LAYOUT TECHNIQUES FOR PHASE CORRECT AND GRIDDED WIRING

By

KEVIN W MCCULLEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2006

For Judy, Shannon, and Sean.

ACKNOWLEDGMENTS

First and foremost, I thank my family; my wife Judy, daughter Shannon, and son Sean. They have shown remarkable patience and understanding. My parents I thank for everything; especially my love of reading, my love of critical thinking, and my lifelong pursuit of knowledge. I thank Professor Dankel for starting me on this path. I thank my advisor, Professor Sahni, for teaching me the craft of research. This was a long process, and there are many other people that I need to thank for their help, patience, and encouragement. I thank my managers at IBM: Dr. Paul Bassett, Patrick O'Connor, Gary Doyle, Douglas Lewellen, and Dr. Robert Allen. All of my team members and coworkers at IBM deserve my thanks for doing more than their share of the work while I pursued my doctorate. Among my coworkers, two deserve special recognition: Robert Walker and Dr. John Cohn. As my team leader and mentor, respectively, Bob and John have helped me grow in immeasurable ways. Finally, I would like to thank Dr. Ann Cutler and Dr. David Overhauser, long time friends and sources of encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

LAYOUT TECHNIQUES FOR PHASE CORRECT AND GRIDDED WIRING

By

Kevin W McCullen

December 2006

Chair: Sartaj Sahni
Major Department: Computer and Information Science and Engineering

The photolithography tools used for semiconductor manufacturing have not advanced in capability as rapidly as new technologies have been introduced. The result is that we are now printing semiconductor chips with feature sizes much smaller than the wavelength of light used by the lithography tools. The gap between light wavelength and feature size has been bridged using Resolution Enhancement Techniques (RET). As more aggressive RET techniques have been applied to semiconductor manufacturing the layout styles used in design have had to adapt to new requirements.

Traditionally, the rules used to define a legal semiconductor layout were local in nature, encompassing restrictions such as minimum spacing between features and minimum feature sizes. The introduction of Alternating Phase Shift Masks (AltPSM) introduces design constraints that are no longer local in nature. In order to achieve design closure (rapid and assured completion of the design), layout restrictions have been proposed for AltPSM. These layout restrictions assure a 'correct-by-construction' layout. These techniques have been applied to the polysilicon gate layer of designs. Future

decreases in minimum feature size will cause these problems to be seen in the metal wiring of semiconductor designs.

This research defines a set of correct-by-construction rules for AltPSM layouts of metal wiring, and provides algorithms and results for three phases of the layout process: automatic wiring, technology migration, and design for manufacturability.

Automatic wiring is the process of connecting the components of a design together by describing the wiring paths that connect each set of connections. Technology migration is the process of converting a design from one technology generation to another, by applying the newer technology's rules and restrictions to the layout. Design for manufacturability describes the addition of features to a layout to improve the yield of the layout in manufacturing. This research shows that each of these can be performed effectively on layouts with restricted topologies.

CHAPTER 1
INTRODUCTION

Introduction

Microprocessors tick to a clock measured in picoseconds. The technologies used to produce microprocessors tick to a clock measured in "nodes." The ITRS (International Technology Roadmap for Semiconductors) [25] defines a series of technology nodes. Each node is commonly named after the metal "half-pitch" (the half-pitch being one half of the center to center spacing for metal lines). In 2006, the industry is moving to production manufacturing of the 65 nm (nanometer) node, with the first test chips (also known as test sites) being produced in the 45 nm node.

With each new technology node, the use of Resolution Enhancement Technology (RET) has become more aggressive. Designers are being exposed to new RET related layout restrictions [26,33]. Two major sources of RET related layout restrictions are the introduction of Alternating Aperture Phase Shift Masks (AltPSM) and "Gridded Layout" styles.

Introduction to Gridded Layout

Gridded Layout styles restrict the legal topologies on the chip to simplify the patterns that need to be imaged by the lithography tools and printed during the manufacturing process. Gridded Layout styles are a form of "Lithography Friendly" design, "To improve manufacturability, it is necessary to limit the configuration of circuit patterns" [59]. Scheffer [53] refers to "Routing to a strict methodology" in describing a gridded type of approach for metal wiring. In Liebmann [40], IBM's "Radical Design

Restrictions" (RDR) are described. One of the key features is consistent orientations for narrow lines:

> "single orientation of narrow features: maintaining a largely parallel and one-dimensional layout environment eliminates all localized RET design conflicts, all but guaranteeing an uncompromised RET solution. While RET-compliance can be facilitiated by enforcing a single orientation over large layout blocks, as illustrated in figure 4, restricting the most line-width critical devices to a single orientation over the entire chip, further improves dimensional control by simplifying the OPC problem and eliminating potentially significant error sources."

In Liebmann et al. [41], there is a "caricature of the ideal layout as a lithographer sees it", an array of evenly spaced, equal width, parallel lines. The description of these lines continues: "performance-simulation, process-optimization, mask-manufacturing, and RET-solutions are well understood for one-dimensional gratings … everything else becomes an approximation."

The RDR layout style is currently applied to the gate orientation. However, the minimum pitch for metal lines in the ITRS follows closely (typically within one technology node) behind the gate pitch. The current node's RET challenges for gate layout become the next node's RET challenges for wiring. In the ITRS[25] section on Design, "Radically-restricted rules (grid-like layouts, no diagonals, etc.)" are shown in research phase during 2005-2007, Development underway from 2007-2010, and in preproduction until 2011.

Introduction to AltPSM

AltPSM was first identified in 1982 [36]. AltPSM is a technique used to improve the minimum feature size that can be printed for a given wavelength of light. AltPSM techniques take advantage of constructive and destructive interference patterns to sharpen edges that are printed on a wafer. A standard photolithography mask is a transparent piece of quartz (commonly referred to as "glass" in the semiconductor industry) with a

layer of material (typically chrome) deposited in patterns to create areas that are opaque. An AltPSM mask has a similar bulk material, but in addition to the chrome layer some transparent sections of the mask are either etched or have an additional transparent layer deposited, changing the depth of transparent material in those sections. Changing the depth of material traversed by a beam of light alters the phase of the light transmitted. By carefully controlling the depth of the transparent material, an AltPSM mask creates areas in which the light passing through the mask has phases that are 180 degrees out of phase. When the light with these opposing phases meets at the wafer, the result is constructive and destructive interference.

There are numerous styles of AltPSM. We will be dealing with Bright Field AltPSM and Dark Field AltPSM. We will concentrate on the application of Dark Field AltPSM to the wiring layers of chips. We use the definitions of Bright and Dark field that are presented by Wang [58].

Lithography Fundamentals

The wavelength of light currently used in semiconductor manufacturing is 193 nm (deep ultraviolet). Resolution enhancement techniques are required in order to print features that are smaller the wavelength used [37,58,59]. Various techniques such as off axis illumination and optical proximity correction have been used to enable the 90 nm node to enter manufacturing. Depending upon the lithography solution used, the 65 nm node requires the use of AltPSM in order to print minimum size features [25,41]. AltPSM techniques are expected to be required for local interconnect metal in the 45 nm node [25], which is now (after a recent revision from 2 year technology introductions to a 3 year cycle) expected to enter manufacturing in 2010. The first 45 nm test sites have already been announced [31].

The Rayleigh equation for optics specifies that the minimum printable feature size S is a function of a constant K_1, the wavelength of light used (λ), and the numerical aperture of the lens (NA) [38]: $S = K_1 \frac{\lambda}{NA}$. In order to improve the resolution of a semiconductor process, one of the three factors in the Rayleigh equation needs to be improved. Making any significant changes to the tooling of a process is expensive. The cost of a single photolithography tool has passed $26,000,000 [32], and building a new semiconductor fabrication facility can cost more than $2.5 billion [62].

The current generation of lithographic tools use 193 nm wavelength light (λ) [38], with a Numerical Aperture (NA) of 0.80 to 0.85 [38,61]. Immersion tools are available with an NA of 1.35 [44]. K_1 is largely a function of the process and RET solution used.

Improving λ is a daunting process, requiring the replacement of virtually the entire lithography process: light sources, lenses, masks, and resists. Difficulties with materials (CaF_2 lenses and resist composition), along with the requirement that the entire optical path be completely free of water vapor or oxygen, have caused 157 nm lithography to drop out of the plans for future technologies [30,38]. Until the advent of EUL (Extreme Ultraviolet) lithography, there is no prospect for improvement in λ in lithographic systems. EUV systems ($\lambda = 11 - 14nm$) pose considerable engineering challenges. Below 100nm "there are no known transparent materials" [38], meaning that EUL optical paths will be reflective, using mirrors rather than lenses to focus the beams. The mirrors in the optical path will require surface smoothness errors of less than 0.25 nm. Masks will also have to be built as selectively reflecting mirrors for EUV.

Numerical Aperture is limited by the size of the lens system and the refractive index of the material between the lens and the mask. Incremental increases in NA require

larger and larger lenses, as an NA of 1.0 (the theoretical limit in air) in theory requires a lens that captures and transmits 100% of the light generated at the source [58]. For an immersion tool with an NA of 1.35, there is very little headway left for improvement using water as an immersion media. The refractive properties of water limit the NA to approximately 1.4, while lens design limits the NA to below 1.35 [42].

The K_1 value captures the ability of the process (lithographic and chemical) to resolve features. K_1 is limited to values greater than 0.25, because at a half-pitch of $\frac{0.25\lambda}{NA}$ the contrast drops to zero and nothing is printed [38]. Phase shift masks make it possible to approach the theoretical limit of $K_1 = 0.25$ [36,38], representing a doubling of the resolution over what is available without aggressive RET.

In Figure 2, also found in a similar form in Berman et al. and Levenson et al. [7,36], we see how the mask with phase shifters (on the right) transmits light with an electric field ("E" in the diagram) that has changed sign, while maintaining the same intensity. When the light reaches the wafer, some spreading has occurred. The intensity of the light reaching the wafer is strongly influenced by the diffraction effects from the two out of phase E peaks. The result is a well defined 'dark spot' between the two shifters. This is an example of "Bright Field" AltPSM.

Figure 3 shows more clearly the destructive interference between two light peaks that are out of phase, and have an intensity that falls off at $\frac{\sin(x)}{x}$. The middle waveform is the summation of the top and bottom waveforms, that are out of phase by $\frac{3\pi}{2}$ radians (270 degrees). At X=0, there is a "dark spot" where the top and bottom waves cancel out (destructive interference).

The light that is transmitted through the mask interacts on wafer with a "resist" layer. The resist is a chemical that is sensitive to light (similar in concept to the process used in film photography). Resists are characterized as a "Positive" or "Negative" resist. A positive resist is softened (becomes soluble) as a result of being exposed to light. A negative resist is initially soluble, but becomes insoluble (hardens) when exposed to light. With a Bright Field AltPSM mask and a negative resist, the dark sections shown in Figure 2 and Figure 3 will be washed away during processing (the surrounding areas will be made insoluble), and gate material can be deposited or built in this region.

Figure 4 shows an example of Bright Field AltPSM. The design shape (the center dark region) is flanked by two phase shapes, one on the left and one on the right. The two phase shapes must have opposite phases in order to properly print the design shape.

Figure 5 shows an actual layout for Bright Field AltPSM. The black shapes are the Polysilicon shapes, with the gates being the narrow sections. The gray shapes are the two sets of phase shifters. One of these colors (which does not matter, nor is that detail visible to the designers) represents sections of the mask that are etched back or built up as shown in Figure 2. The black shapes are covered by chrome in Figure 2. The actual phase assignments are invisible to the designer, and they may change (or flip) as a model is placed within the hierarchy of a chip. If a model is used twice in a design, the phases assigned may differ in each placement, depending upon the neighborhood and phase shapes at the cell boundaries.

Bright Field AltPSM was initially the most widely studied form of AltPSM. These techniques have been used by memory designers for many years. Designers would build early hardware using AltPSM techniques so that they could use the previous generation's

process, tooling, and lithography in order to get early hardware experience. By the time the designs were ready to enter volume manufacturing, the process and lithography had improved to the point that AltPSM was no longer required, and standard layout techniques were used. With the 65 nm node, immersion lithography became available at a critical time, and the need for AltPSM was dodged yet again. With the 45 nm ITRS node there is little hope that a similar breakthrough will occur, and AltPSM may become the plan of record for semiconductor manufacturing. EUV lithography is currently targeted for nodes past the 32 nm node (production in 2013) [25].

With Bright Field AltPSM, the mask shapes are derived from the design shapes. For designers, this creates confusion due to the fact that there are layout restrictions (and ground rules) that are imposed as the result of shapes that they do not create. For both tools and designers, a correct layout is no longer a function of only local (edge to edge) distances. Non-local shape topologies can create illegal layouts.

Dark Field AltPSM is a complement (or dual) of Bright Field AltPSM. In Bright Field AltPSM, we used phase shifters to create the design shapes. In Dark Field AltPSM, we assign phases to the design shapes themselves, in order to sharpen the focus of the empty spaces between the shapes. Another way of looking at this is that Bright Field AltPSM's critical feature (the feature that is hard to print and control) is the design shapes, typically the actual gate layer. Dark Field's critical feature is the space between design shapes. In Bright Field we are attempting to control the critical dimension of the gates (the distance across the shape) in order to better control the performance and function of the circuits. In Dark Field, we are attempting to control the spacing between lines, in order to improve the wiring density and the yield of the chip.

Figure 6 shows an example of Dark Field AltPSM. The design shapes (three wires in this case) are each assigned phases. The destructive interference between the design shapes sharpens the contrast to the level required in order to create well defined spaces between the wires [58,59].

Both Bright and Dark Field AltPSM require that the phase shapes be colored in a way which guarantees that all critical features are flanked by opposing phases. The phases must alternate across the critical features. The coloring problem creates topological constraints. There are shape topologies that cannot be colored, meaning that the alternating phase requirement cannot be met.

The definition of Bright Field AltPSM (Alternating Aperture AltPSM for Polysilicon layers) is relatively settled. Technology ground rule manuals that include AltPSM for gate layers have been written. The exact details of Dark Field AltPSM usage in metal wiring layers are still unsettled and the exact implementation details may differ somewhat. However the concept remains that Dark Field AltPSM imposes graph colorability constraints on the metal wiring layers of a chip layout. These layers have commonly been designed using automated wiring programs, making it important to understand the implications of graph colorability constraints on wiring algorithms and layout methodologies.

This research defines a strictly gridded layout style that guarantees Dark Field phase colorability. We then apply this layout style to the problems of routing, technology migration, and design for manufacturability.

Phase Conflicts in AltPSM Layouts

Leung [35] observed in a footnote that ground rule restrictions may suffice in dealing with some RET details. However, the removal of phase conflicts during design has been found to be impractical. In Liebmann et al. [41] it is stated that:

> "Exhaustive investigation of altPSM layout-rules, -tools, and –methodologies (12) resulted in the conclusion that altPSM layout restrictions, with reasonable impact on existing layouts, could not be communicated through conventional design rules but required the integration of altPSM design capability throughout the physical design flow."

Work has been done in the area of defining methods for producing phase-correct results for custom layout, primarily with respect to transistors, in the context of Bright Field AltPSM [7,8,23,27,28].

In graph theoretic terms, every AltPSM layout produces a "Phase Conflict" graph, which must be two colored. Every shape with a phase color is represented by a node in this graph. In Bright Field, the nodes represent the phase shapes, in Dark Field the nodes represent design shapes. There is an arc between every pair of nodes representing shapes which are a critical distance apart (typically the minimum shape width in Bright Field, or the minimum shape separation in Dark Field). In order to assign phase colors in a way which guarantees that no critical features have the same color on either side, this graph must be Two-Colorable, also known as Bipartite.

The MAXCUT Problem

For a graph to be bipartite, it is necessary that there be no odd-cycles in the graph [6]. An odd-cycle is any closed path within the graph which contains an odd-number of nodes. The problem of taking a graph and making it bipartite involves taking the MAXCUT of the graph. MAXCUT produces a cut of the graph with the maximum weight, dividing the graph into two sets of nodes such that the weight of the edges that

pass between these sets is maximal. Removing the edges that do not lie in the cut produced by MAXCUT produces a graph which is maximally bipartite (adding any edge back into the graph will leave the graph non-bipartite).

In Figure 6, the MAXCUT is the set of edges which connect the region on the left with the region on the right. The edge(s) which do not fall in the cut (one edge in the region on the right) can be removed to make the graph bipartite.

The general MAXCUT problem is NP-complete [16]. However, there is a polynomial time solution for MAXCUT on planar graphs [20,49]. There are also a number of approximation algorithms for MAXCUT.

For planar graphs, the efficient implementation of MAXCUT is done via the planar dual graph. This is the key feature which differentiates the planar graph solution from the NP-complete problem in general graphs. In the planar dual, each cycle in the primal graph is represented as a node in the dual graph. The degree of each dual node is the size of the cycle it represents in the primal. Hence, each odd-cycle in the primal graph becomes an odd-degree vertex in the dual. The MAXCUT of a planar graph is the minimum odd-vertex pairing in the planar dual [20,49]. Finding the minimum odd-vertex pairing is a maximum matching problem. Hadlock [20] refers to Edmonds [14] for a polynomial time solution to the matching problem. Edmonds' algorithm requires $O(v^4)$ time and $O(v^2)$ memory (which is roughly equivalent to the memory needed to store a dense graph). More efficient algorithms for matching have been found, with the best in the neighborhood of Gabow and Tarjan [15].

Berman et al. establish the planarity of the phase conflict graph for Bright Field AltPSM through the following proof [8]:

Theorem 1 *Given the following conditions, the Phase Conflict Graph is Planar:*

1. *Let the minimum shape width be w, the minimum shape to shape spacing be b, and B be the distance between two shapes which guarantees no phase conflict (if two shapes are spaced further than B from each other, there is no arc present in the phase conflict graph).*
2. *Let* $w \geq b$ *and* $B \geq 2b$
3. *Four rectangles that are pairwise in conflict have no diagonal conflicts.*

Represent each shape with a node located within the shape: N_s. For each arc in the conflict graph, replace the arc from N_1 to N_2 with three arcs:

1. One arc runs from N_1 to the boundary of N_1 nearest N_2.
2. One arc runs from N_2 to the boundary of N_2 nearest N_1.
3. The third arc connects these two new arcs.

The internal arcs can be routed without intersections (they run from a central node within the shape to the edges of the shape). Each external arc must be shorter than *B*, therefore they cannot enter a shape and then leave the shape. They must be shorter than *B* because we have defined *B* to be the minimum distance which creates a phase conflict. Therefore any intersection must occur outside the shapes.

Now let there be two outside segments (from 2.c above) which intersect. Call them (a,b) and (c,d). Assume that a and c are on the same shape (the paper specifies that a and b are on the same shape, however a and c makes sense). Since (a,b) and (c,d) intersect, $|a,b|+|c,d|>|a,d|+|b,c|$. Therefore, either (a,b) is longer than (a,d) or (c,d) is longer than (c,b). This contradicts condition 2.

If a, b, c, and d are all on separate shapes, then the pairwise distances (other than (a,b) and (c,d)) are smaller than *B*, which contradicts the condition that there be no diagonal conflicts. ■

This argument applies equally well to Dark Field AltPSM. The phase colorings are applied to a set of shapes which are arranged in a planar manner, and which obey the same basic coloring constraints. For our purposes, the key difference between a Dark Field and Bright Field phase conflict graph is the difference in how design shapes must be modified in order to remove an arc from the graph.

There are a number of randomized methods for MAXCUT. A randomized algorithm is characterized by a performance ratio as well as the algorithmic performance. The performance ratio tells how close the achieved value is guaranteed to be to the optimum value. An exact algorithm will achieve a performance ratio of 1.0.

In [52], a performance 0.5 algorithm is presented for MAXCUT. For MAXCUT with two sets S1 and S2, this algorithm consists of placing the first vertex into S1. Each additional vertex is then placed into whichever of S1 and S2 results in a lower internal weight (sum of the weights of edges within the set).

Cho et al. [10] present an approximation algorithm for MAXCUT, called Simmax II. Simmax II operates in order O(v + e), and guarantees a performance ratio of .5 + 1/(2v). While the runtime (linear) of Simmax II is very compelling, the performance guarantee for large graphs is not so compelling. For large numbers of vertices, the performance guarantee does not rise significantly above 0.5.

Another approximation algorithm for MAXCUT is the "Semi-Definite " relaxation [18]. In the Semi-Definite relaxation, MAXCUT is solved by posing the problem as an integer quadratic problem (which is also NP-complete), obtaining a relaxation of this problem, and then solving the relaxation via Semi-Definite programming.

The MAXCUT problem is first posed as an integer quadratic problem:

Given a graph with vertices: $V = \{1..n\}$ and non-negative weights $w_{ij} = w_{ji} \forall i, j \in V$:

Maximize: $\frac{1}{2}\sum_{i<j} w_{ij}\left(1 - y_i \bullet y_j\right)$

Subject to: $y_i \in \{-1,1\} \forall i \in V$

This problem is solved using semi-definite programming, to find an "optimal set of vectors v_i". They then chose a random vector in S_n, and use that vector to define a hyperplane which bisects S_n. All of the original vectors which lie "above" this plane (based upon the sign of their dot-product with the random vector) are assigned to one side of the cut, with the remaining vectors assigned to the opposing side of the cut.

This innovative relaxation of MAXCUT achieves a performance ratio of .879. The runtime for semi-definite programs is polynomial, if there is a known bound on the solution size [1].

Two further approaches to MAXCUT, using the T-Join on planar graphs, and using Voronoi diagrams, will be discussed in the section on prior work.

Bright Field AltPSM Phase Conflicts

In the figures which follow, the nodes and arcs in the conflict graph are drawn over the design and phase shapes, to illustrate the relationship between design and the conflict graph.

Bright Field conflicts are caused by design shapes, or groups of shapes, which induce phase shapes that cannot be 2-colored. There are several typical types of Bright Field phase conflicts and methods for breaking those conflicts [23,28].

The "T" shape conflict, shown in Figure 7, consists of a critical width shape with a "T" shape. The "T" creates a 3-cycle in the phase conflict graph as shown in the figure.

An "Odd-Even" conflict, shown in Figure 8, occurs when an odd number of critical wires with minimum spacings become an even number of critical wires with minimum spacing, through the introduction of a wire end and a jog.

A "Line-End" conflict, shown in Figure 9, is caused by two orthogonal critical wires which are placed closely enough to cause a phase conflict.

Dark Field AltPSM Phase Conflicts

Dark field conflicts occur when the unioned design shapes which are within the critical distance of each other cannot be 2-colored. The space between these shapes is the critical region, known as a critical space [58].

In Figure 10, we see a Dark Field phase conflict which is caused by two wires ending too closely together, while a third wire passes the gap at minimum spacing. In this case, there is a "T" shaped space created, with the base of the "T" being the gap between the wires.

Figure 11 shows the Dark Field equivalent of an "Odd-Even" run. In this case, the phase conflict is caused by the requirement that the top wire (the wire which bends) must carry the same phase for its entire length. This creates the 3-cycle shown, since the top wire cannot have an alternating phase from both the middle wire and the bottom wire, if the middle wire and bottom wire also have differing phases.

Prior Work on AltPSM Phase Conflict Removal

Previous work in the area of AltPSM layout initially (in the early to mid 1990's) concentrated on what we call "Dark Field" AltPSM, but has since concentrated solely on the Bright Field AltPSM problem for Polysilicon layers. Until very recently, all of the previous work has concentrated on the layout fixup problem. These are techniques for taking a completed layout and attempting to fix coloring errors in a manner which is least disruptive to the designer's intent, or for checking a completed layout to determine if it is phase colorable (with the actual fixup of errors left to the designer).

In a paper by Moniwa et al. [45], the removal of phase conflicts is prioritized. The features of the layout are assigned priorities based upon how difficult it would be to make modifications. They identify layers where there is little margin for alignment from layer to layer, where there are features with size restrictions, where there are very narrow phase apertures, or where two phases have a very long common edge. These regions are assigned a priority, and the phases are assigned in that order. The actual phase assignment and fixup in this process is manual. This paper addressed the issue of Dark Field AltPSM, with potential extensions of the idea to Bright Field. In another paper [46], this same group addressed the issue of conflict removal directly on the phase conflict graph. They identify the odd-cycles in the phase conflict graph, and use a heuristic to choose which arcs to remove. Their heuristic is to select the odd-cycle arcs which appear most often in the smallest cycles. They then remove these arcs by widening the spacing between shapes represented by the nodes connected by these arcs.

In Ooi et al. [47,48], two methods for inserting phase information are identified. They look at the problem which we refer to as Dark Field AltPSM, in that they are assigning phases to design shapes. In the first method, layout is performed manually and

an interactive tool identifies the phase conflict errors. The designer is shown the phase conflict graph, and is then free to move shapes in order to remove the conflict(s). In the second method, layout compaction [54] is used to correct errors automatically. They assign phases to the phase shapes, and then proceed to compact the design using rules which preserve the relationship between opposing phases and same phases. Opposing phases may be spaced at the minimum spacing, while same phases must maintain a wider separation.

In the late 1990s, Andrew Kahng's group at UCLA took up the issue of Bright Field AltPSM layout. Their work has generated a number of papers and several new algorithms for generating a bipartite phase conflict graph. The primary contribution by this group has been the introduction of new methods for removing phase conflict, using heuristics, the T-Join, and Voronoi diagrams. The T-Join algorithm is an exact algorithm, while the Voronoi algorithm is an approximation algorithm.

The Optimal T-Join problem is stated [8] as: "Given a graph G with weighted edges, and a subset of nodes T, find the minimum width edge set A such that a node u is incident to an odd number of edges of A if and only if $u \in T$." In another paper [11], it is shown that for a planar graph, solving the Optimal T-Join problem with an empty T is equivalent to solving the MAXCUT. In Berman et al. [8], the Optimal T-Join in the planar dual [of the conflict graph] is solved by replacing some of the nodes with gadgets, and then performing a Minimum-Weight Perfect Matching on the transformed graph. To simplify the matching problem, all of the nodes with degree greater than one are replaced with a set of nodes which are connected by zero weight edges. The result of this transformation is a graph in which each node only touches one non-zero weight arc,

which simplifies the matching. The performance of this algorithm is given as:

$O\left(\left(n\log n\right)^{\frac{3}{2}}\right)\alpha\left(n\right)$, where $\alpha\left(n\right)$ is the inverse of the Ackerman function.

The approximate algorithm in Berman et al. [8] uses Voronoi diagrams in the planar dual. A Voronoi diagram [12] is a space division in which each face contains all of the area which is closer to the central node of that face than to any other central node in the graph. For example, Voronoi diagram of a city with respect to Post Office locations would place each house into the region in which it is closer to that region's Post Office than to any other Post Office. In the algorithm in Berman et al. [8], the planar dual is built, and the odd-degree vertices are identified. A Voronoi diagram is built such that each region is centered upon an odd-degree vertex, and contains all of the even-degree vertices which are closer (by path length) to that odd-degree vertex. They then build a dual of the Voronoi, where each vertex represents a Voronoi region, with an arc connecting each pair of vertices which have a shortest path connecting their centers which is contained within their two regions. A minimum weight matching is performed on the Voronoi dual, and the edges in the planar dual which correspond to the matching are deleted.

Kahng et al. [27] presented four methods for resolving phase conflicts, two of which were seen in Berman et al. [8]. In this paper, they take up the issue of Node-Deletion, as opposed to the Edge-Deletion approaches used elsewhere. The node deletion approach to MAXCUT remains NP-hard, even for planar graphs. The two heuristics implemented are:

1. Goemens and Williamson Planar-Dual approximation [19].

   This algorithm is not the Semi-definite MAXCUT algorithm by the same

authors. This algorithm is an $O(V^2)$ node-deletion algorithm which provides a result which is no more than 9/4 worse than optimum. In this algorithm we inspect the odd faces of the graph. We refer to an iteration of this algorithm as an "age". Each odd face has an age which tells how many iterations it has been in existence. The weight of each vertex is the sum of the ages of all faces that contain the vertex. Iterating until there are no odd faces remaining, remove the node with the highest weight and place it in a LIFO queue. If this creates a new odd face, set the new face's age to 0. Now remove nodes from the queue and add them back. If this vertex creates an odd face when re-inserted, remove this vertex (permanently).

2. Greedy Vertex-Covering Algorithm.

   This algorithm colors the vertices using a breadth-first search. We can then define the "violating degree" of a vertex as the number of edges adjacent to this vertex which connect two vertices of the same color. Processing the vertices in decreasing order of violating degree, remove nodes until there are no violating vertices remaining in the graph. The runtime of this algorithm is O(V log V) (controlled by the time required to sort the vertices by violating degree). There is no performance guarantee for this algorithm.

Correct by Construction Layout

Design-based solutions to avoiding phase conflicts are driven by several concerns:

1. The general difficulty of merging MAXCUT implementations with CAD algorithms.
2. The difficulty of managing the creation of phase shapes in a completed chip layout [39,57]. The design layout is hierarchical, while the phase shapes are

a mask feature, and are added to the design flat. The creation of phase shapes cannot be done hierarchically, because a sub-cell that is used in multiple environments may need to have different phase shape arrangements or phase colorings added to each instance based upon the phase shapes and colorings of neighboring cells.

3. The phase shift creation and phase color assignment cannot be done until the design is substantially complete (due to cell-to-cell interactions). However, because of the non-local nature of potential phase conflicts, phase shifting may require design changes that extend beyond a single cell. There is no guarantee of this process converging to a completed design.

The difficulties in guaranteeing the ability to phase shift a layout at the time of tapeout (transmission of the completed design to manufacturing) in a predictable amount of time, with a predictable amount of effort, leads directly to gridded layout styles. Gridded layout styles guarantee correct-by-construction phase shifting and phase coloring.

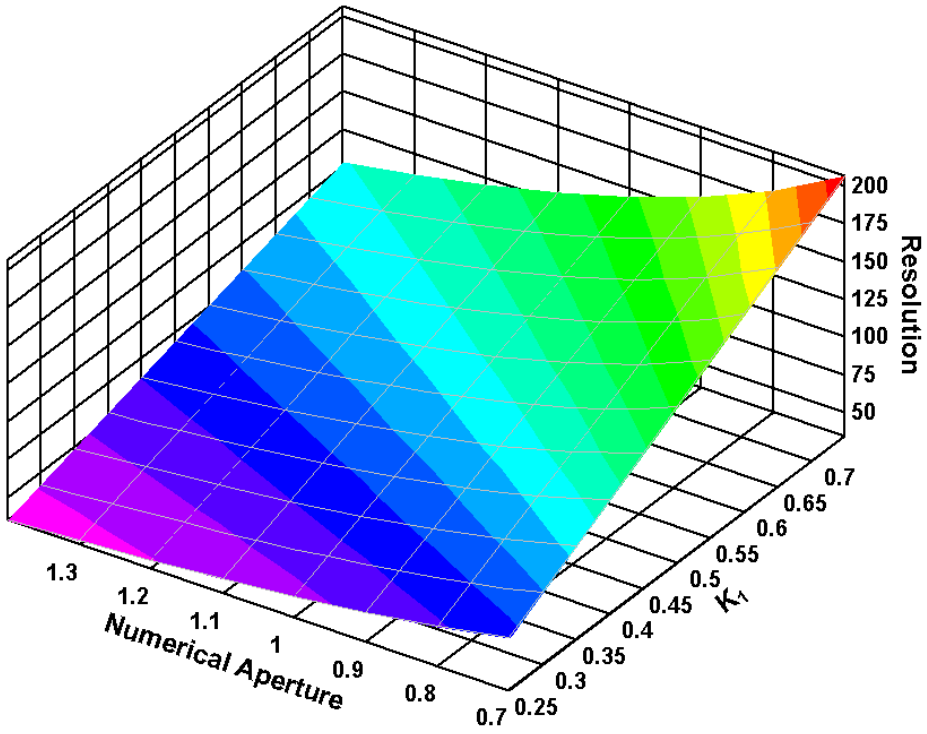

Figure 1. Resolution in nanometers as a function of K_1 and Numerical Aperture for 193 nanometer lithography.

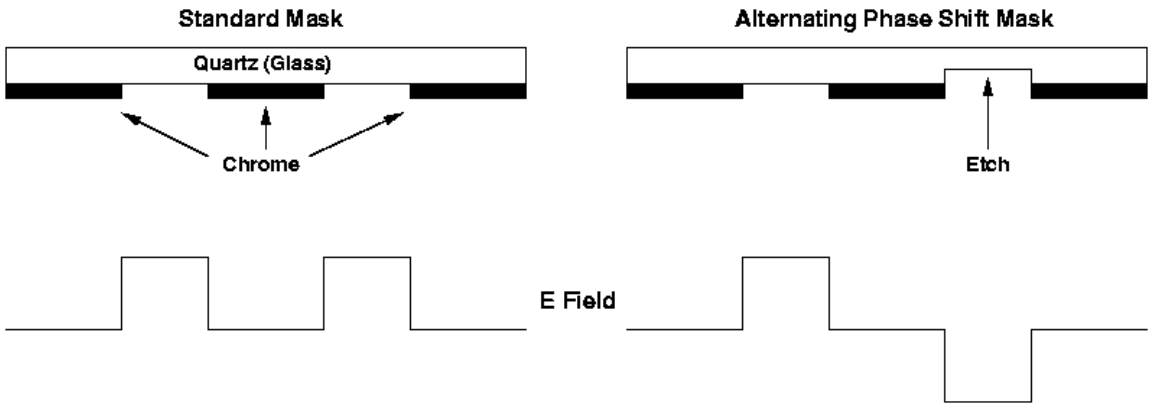

Figure 2. Alternating Phase Shift Mask

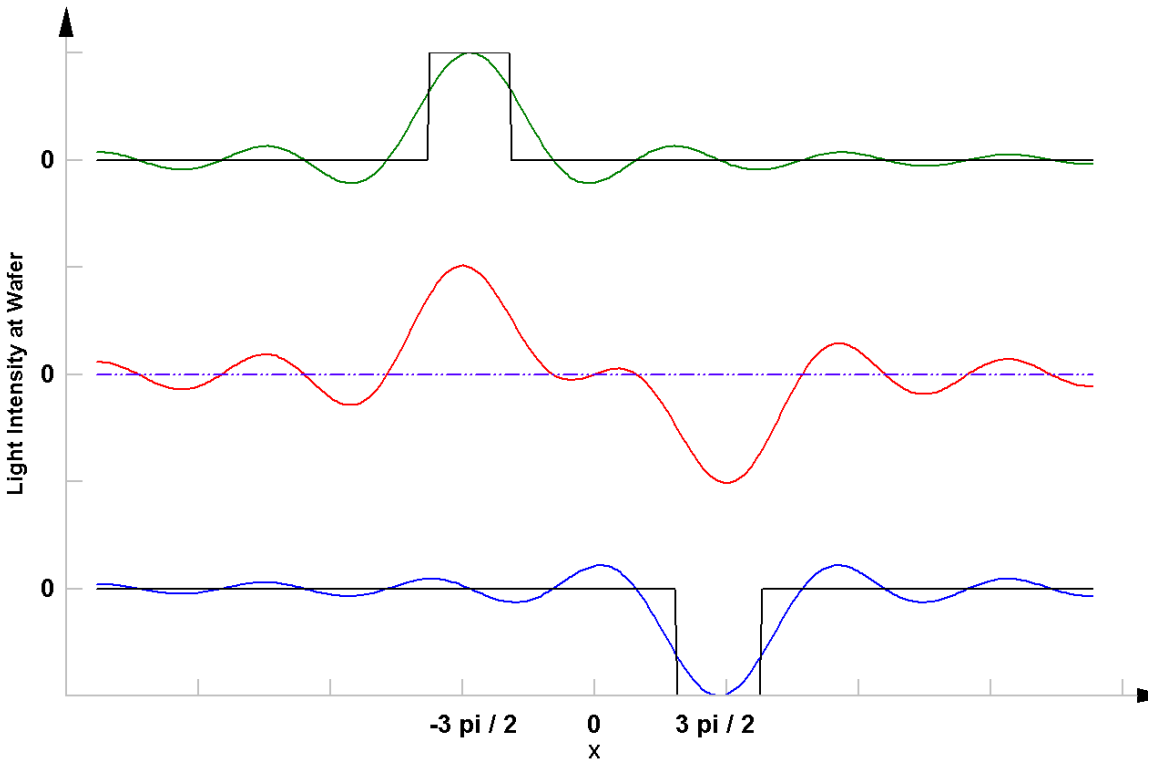

Figure 3. Destructive interference from out of phase light peaks.

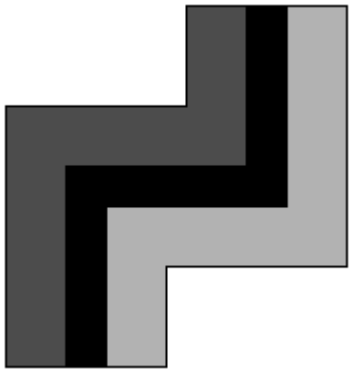

Figure 4. The bright field AltPSM shapes for a simple layout.

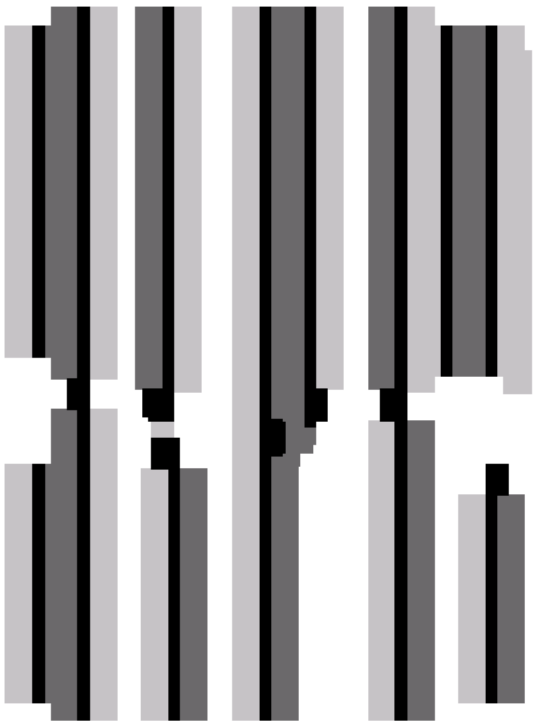

Figure 5. The bright field AltPSM shapes for an actual gate layout.

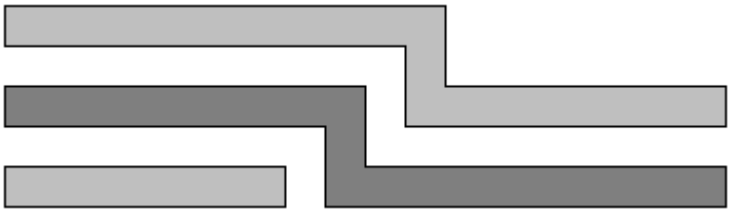

Figure 6. Example wires phase colored for dark field AltPSM.

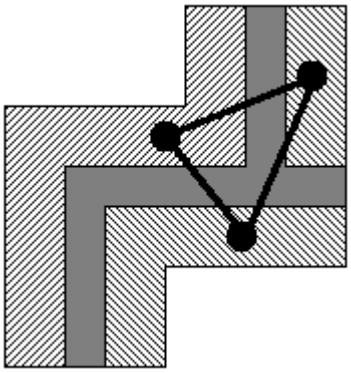

Figure 7. The bright field AltPSM "T" conflict.

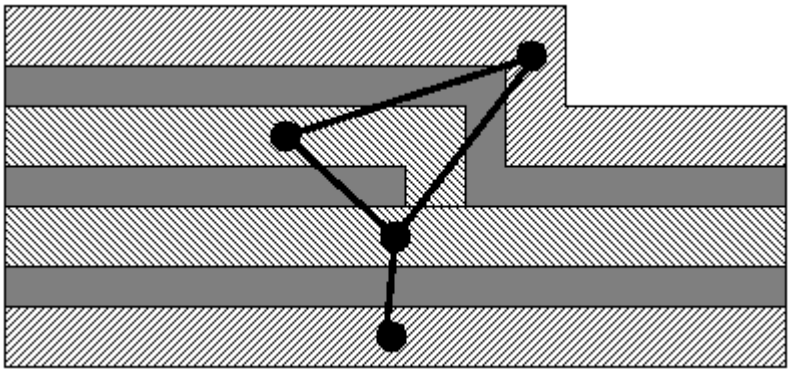

Figure 8. The bright field AltPSM "Odd-Even" conflict.

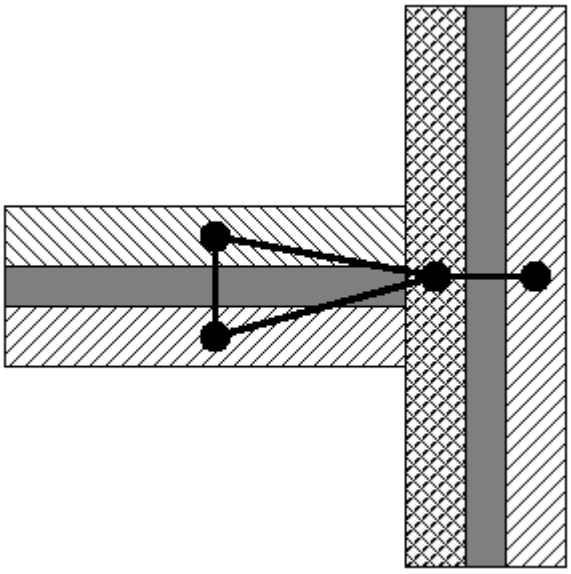

Figure 9. The bright field AltPSM "Line End" conflict.

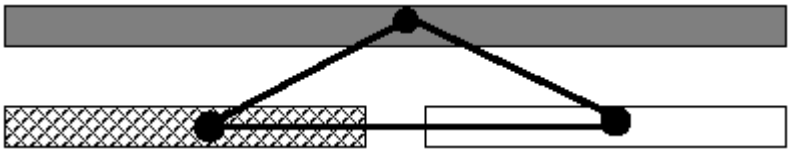

Figure 10. The dark field AltPSM "T" conflict.

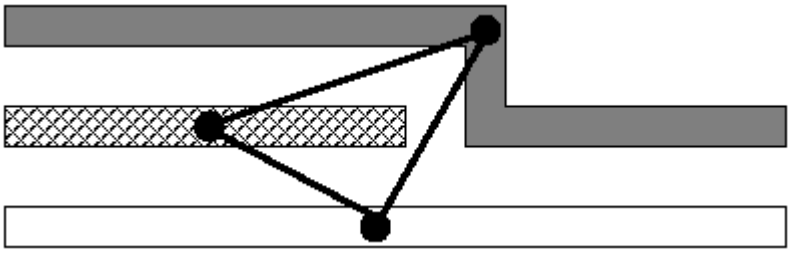

Figure 11. The dark field AltPSM "Odd-Even" conflict.

Table 1. Guaranteed Performance of the SIMMAX II Algorithm.

Nodes	Performance Guarantee
2	0.75
5	0.60
10	0.55
50	0.51
100	0.505

CHAPTER 2
LAYOUT RESTRICTIONS

Our research investigates the feasibility of performing Phase Correct and Gridded layout for routing layers. Previous work has concentrated on the gate layers of chips [39,57].

The layout restrictions that we propose are all local in nature, yet they lead to a globally phase colorable solution. These restrictions have the effect of making the layout more “conservative” in nature, in that potential phase conflicts are avoided. This lowers the layout density slightly, due to the fact that certain topologies are forbidden.

In the following, we primarily concentrate on the Dark Field routing problem. Bright Field phase shift masks are primarily used in the gate layers of designs. Automated routing can be performed in that context, but in most cases the gate layers are laid out by hand or by using parameterized cell generators or libraries. As little routing is typically done on the gate layer as possible.

Bright Field AltPSM Routing Restrictions

For Bright Field routing, we require two restrictions:

1. All wires in the primary wiring direction must be on a uniform wiring pitch and may be critical or non-critical. This allows phase shapes to be laid out in parallel stripes.
2. All wires running orthogonal to the primary wiring direction (also known as "wrong-way") must be non-critical. This prevents “T”, “Odd-Even”, and

"Line-End" conflicts, by preventing phase relationships from following wrong-way wires.

These restrictions are similar to design restrictions that have been proposed in order to create correct by construction gate layouts [40] and for technology migration [23]. These restrictions allow us to create a phase conflict graph that contains no odd cycles.

In the case of the "T", a phase coloring error is prevented by restriction 2, that requires one leg of every bent piece of wire to be non-critical, as shown in Figure 12. One of the sections of the "T", the leg or top must be non-critical.

The "Odd-Even" conflict is also prevented by restriction 2. The requirement that the wrong-way wire be non-critical breaks the common phase requirement along the top edge. This allows the critical stripes to be colored in rows, with each row having one consistent color along it's entire length, even if the row is broken (as in the center row of the odd-even). The correction of an Odd/Even conflict is shown in Figure 13.

A line-end conflict is averted by restriction 2. Restriction 2 requires that if we have two pieces of wire that are perpendicular, one of them must be non-critical. The correction for a Line End conflict is shown in Figure 14.

Restriction 1, which requires that all critical width wires be in a common direction on a common pitch, allows us to phase color the layout in parallel stripes. Restriction 2, which requires that all wires oriented perpendicular to the critical wires be non-critical, allows us to break the phase relationship along every edge that jogs (changes direction by 90 degrees). Restriction 2 thereby prevents a phase from one stripe from bending into the track of another stripe.

Dark Field AltPSM Routing Restrictions

For Dark Field routing, we require the following three wiring restrictions:

1. All wiring must run in the primary wiring direction, with a uniform spacing or pitch. Wires running orthogonal to the primary wiring direction are forbidden.
2. Any location where a wire ends must have an extra space inserted beyond the wire end, effectively doubling the free space between the end of one wire and the end of the next wire.
3. Pins cannot be aligned in the primary wiring direction (for the pin layer), at minimum spacing. Two such pins create a minimum width spacing, violating restriction 2.

On a Dark Field mask, a "T" conflict is caused by a minimum width gap between two wire ends, where there is a third wire running continuously in either of the tracks adjacent to the gap. In Figure 15, we show how restriction 2 prevents a "T" shaped gap by widening the distance between two line ends, thereby making the space between the line ends non-critical.

Correcting or preventing an Odd-Even run in a Dark Field mask is achieved by prohibiting any wrong-way wires . By preventing any wrong way wires, we prevent a critical edge from ever occupying more than one phase stripe. In the example shown in Figure 16, we break the phase conflict shown in the top figure by moving the wiring jog to a separate wiring layer that runs orthogonal to the primary wiring direction of the non-jogged portions of the wire.

There is an additional complication for Dark Field masks, pin access. If we have two or more pins on minimum pitch, aligned in the primary routing direction, they may

not be accessible. Whether the pins can be routed or they create an intrinsic phase conflict will depend upon the details of how Dark Field masks are implemented. In the worst case via arrays may not be possible depending upon whether the via layer is phase shifted. Two pins aligned in the primary routing direction would require wires that terminate at the pins, leaving a minimum size space. This wire configuration violates restriction number 2.

Correctness

In both the Bright and Dark Field implementations, we create a layout in which the phase shapes are generated in stripes running parallel to the primary wiring direction. The phases in each of these stripes are consistent across the wiring region. We prevent any topologies that cause the phase in a stripe to change, such as a critical edge that jogs (changes direction by 90 degrees). Another way of saying this is that all critical features are designed parallel and on a uniform pitch or periodicity.

Theorem 2 *The Dark Field AltPSM routing restrictions guarantee a phase correct layout.*

We observe the following: The phase conflict graph is planar as proven by Berman et al. [7]. Only nearest neighbor shapes can have phase interactions. Phase conflict arcs do not extend through or past one shape to more distant shapes. A phase conflict arc only exists where two phase colored shapes are within a critical distance of each other.

For Bright Field AltPSM, the critical distance between the two phase shapes is filled by the Polysilicon shape being drawn. For Dark Field AltPSM, the critical distance between the two phase colored wires is the empty space between the wires.

Lemma 1 *Restriction 1 in each case requires that all phase colored shapes are parallel and run in stripes.*

Each stripe has a consistent phase color along its entire length regardless of spaces within the stripe. The pitch for wires and spaces is based upon the critical space or critical wire size. Any wires that span more than one pitch (for Bright Field) are non-critical. Any spaces that span more than one pitch (for Dark Field) are non-critical.

Lemma 2 *Restriction 2 for Bright Field layout, and restrictions 2 and 3 for Dark Field layouts prevent any phase conflict arcs from running parallel to the phase colored shapes.*

In Bright Field AltPSM, all wires that run perpendicular to the primary wiring direction are non-critical, and no phase conflict arc crosses them. In Dark Field AltPSM, all spaces between wires or pins along a stripe are non-critical, and no phase conflict arc spans the space.

From Lemma 2, we conclude that all of the phase conflict arcs flow perpendicular to the phase colored shapes in both Bright and Dark Field wiring. No phase conflict arc flows along a phase colored stripe. In order for there to be a coloring error, there must be an odd-cycle present in the phase conflict graph. In order for there to be an odd-cycle present in this phase conflict graph, it is necessary for there to be two paths moving vertically (for horizontal phase stripes) through the graph that begin at one node and subsequently rejoin, and that have path lengths that differ by an odd number of arcs. It is this configuration that is prohibited by Lemma 1. Because of the fact that we cannot have phase conflict arcs that pass through another shape, or that span a non-critical space, in order for the paths to rejoin, the path lengths must be equal. With two paths of equal length, a head node and a tail node, all cycles in the graph will be even-cycles. All nodes in the conflict graph that are not members of even-cycles will be leaf nodes. Therefore

the phase conflict graph will be two colorable and the design is phase colorable by construction. ■

In Figure 17, the flow of phase conflicts downward (for a horizontally oriented Dark Field layer) is shown. The solid lines show actual phase relationships, while the dashed lines show phase relationships that are forbidden (or prevented) by the phase correct routing rules.

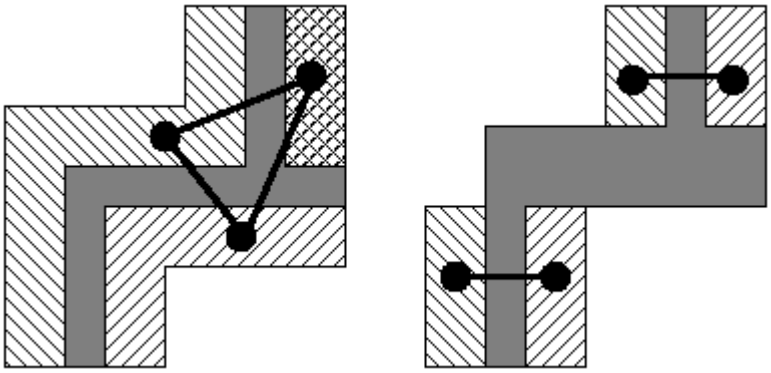

Figure 12. Bright Field AltPSM “T” Conflict Fix

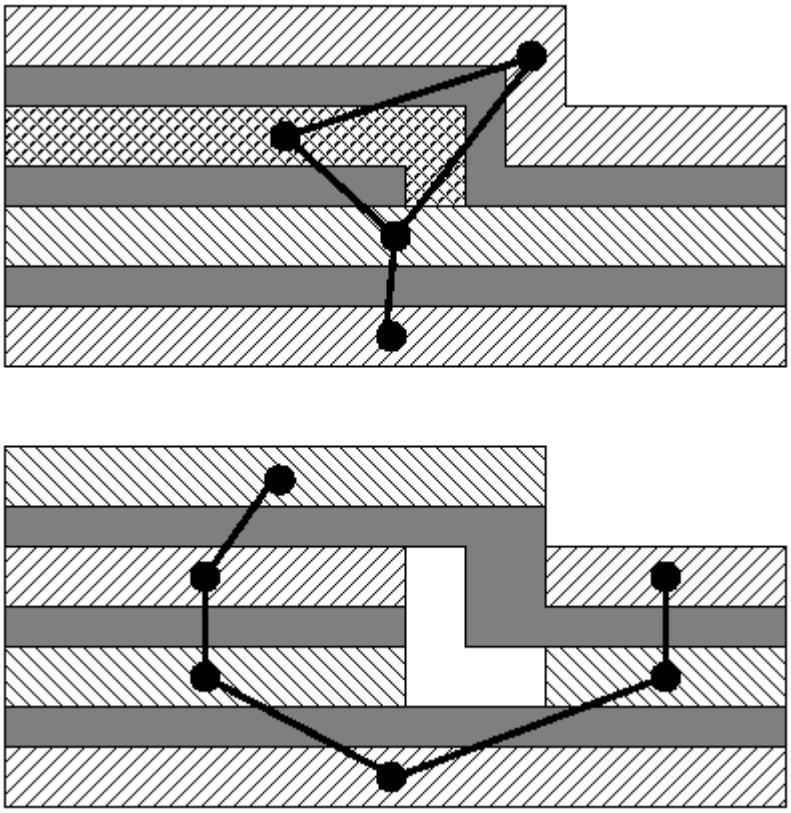

Figure 13. Bright Field AltPSM Odd-Even Fix.

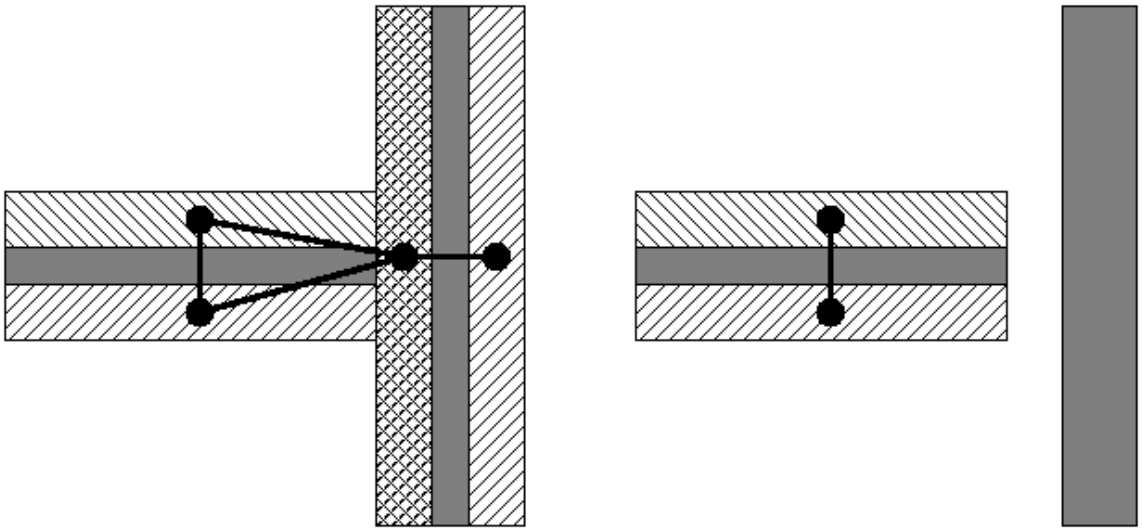

Figure 14. Bright Field AltPSM Line-End Fix.

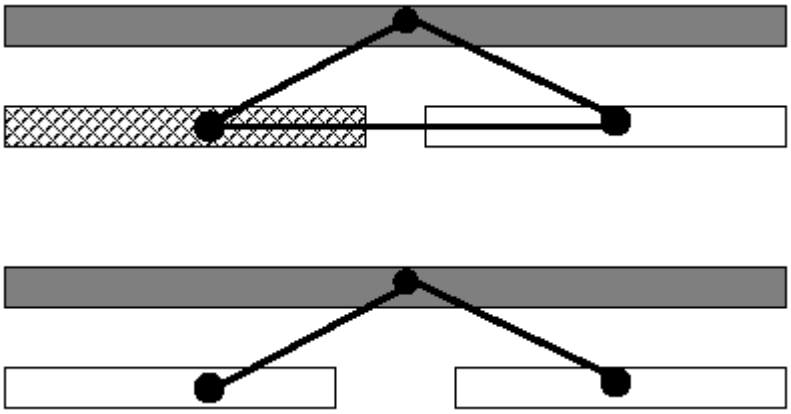

Figure 15. Dark Field AltPSM "T" Conflict Fix.

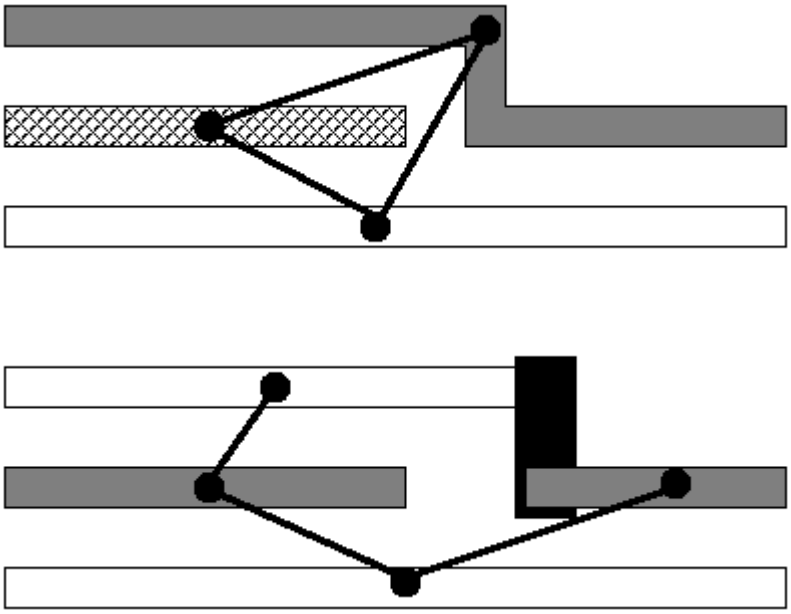

Figure 16. Dark Field AltPSM Odd-Even Run Fix.

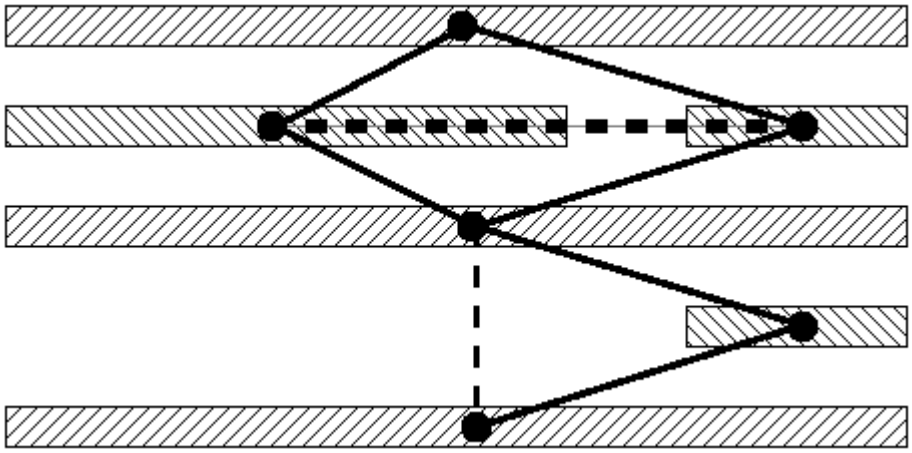

Figure 17. Valid and Invalid Phase Conflicts in a Phase Correct Layout.

CHAPTER 3
ROUTING

Introduction

The function of a router is to correctly wire the connections at the cell, macro or chip level, while taking into account a multitude of objectives [35,54]:

- Completion percentage: routing as many paths as possible.
- Ground rule correctness: routing nets so that no technology ground rules are violated.
- Timing: routing critical nets by a path that does not exceed the wire length budgeted for the route by timing tools.
- Cross-talk: routing nets so that two nets do not have long parallel runs that can cause a signal from one net to influence the signal on the neighboring net.
- Yield: routing nets so that yield is maximized, by using relaxed spacings where possible, and by adding additional vias where possible.

The general routing problem is quite difficult, combining an NP-hard problem [24] with a large number of objectives, some of which are in conflict. For example, timing objectives try to route wires by as direct a route as possible, while congestion, yield, and cross-talk objectives try to push wires further apart.

A large number of variants of routing problems and approaches exist. We look at gridded routers, as applied to switchboxes and macro layouts.

Implementation

The implementation of these algorithms was prototyped in C++ within an existing interactive layout system. The base algorithm used is a gridded multilayer router using best first search [54]. The major difference between the base algorithm and the phase correct algorithm lies in choosing the moves that can be explored. The restrictions noted earlier are implemented as limitations in the search phase of the algorithm. During the retrace stage, the blockages on extra grids are inserted. For a Bright Field wire that is routed perpendicular to the preferred direction, a double width wire is inserted and two side-by-side grid points are blocked at each point along the wires length. For a Dark Field wire, a blocked grid point is placed on the grid that lies one beyond each end of a wire, in the preferred routing direction (a horizontal wire has a blocked grid at the lowest X coordinate of the wire minus 1 and the highest X coordinate of the wire plus 1, provided that the wire does not begin or end on the boundary).

Router Algorithm.

Key differences from a standard maze router are highlighted in bold.

1. Initialize routing costs from the input file.
2. Initialize net and pin list from the input file.
3. Routing Grid $\leftarrow \varnothing$
4. Owner Grid $\leftarrow \varnothing$
5. For each pin, verify that the pin:
    a. **Is not too close to another pin (illegal pin locations)**
    b. **Is not placed inside a blockage (due to roundoff errors in data creation). If inside blockage, attempt to move the pin to a nearby legal location (search 2 grids in each direction)**

c. Is not trapped by blockages. Run a scanline in the primary wiring direction. Stop when a blockage, pin, or escape is found.

6. For each pin, mark the neighboring grid points for pin access scoring.

7. For Each Net:

    a. Select an Unrouted Pin. If two pins have already been connected, only allow pin to net connections (not pin to pin).

    b. Path Trace $\leftarrow \varnothing$

    c. Front heap $\leftarrow \varnothing$

    d. Lowest Cost Grid $\leftarrow \infty$

    e. Add the pin location to the heap of fronts, with cost equal to zero.

    f. While Front Heap $\neq 0$ and no path exists and interations < maximum iterations

        i. Front $\leftarrow$ top of Front Heap (lowest cost entry)

        ii. For each possible neighbor point [there are 6: up, down, left, right, up level, down level]. **For Dark Field, only neighbors in the primary wiring direction are considered.**

            1. Does the Neighbor point exist and is this neighbor point one of the following?

                a. Open: Routing Grid[neighbor] = empty

                b. A target for this net (i.e. a pin for this net): Routing Grid[neighbor] = pin on this net.

                c. For Dark Field: additional grid space is available if we are changing levels.

            2. Move cost $\leftarrow$ front cost + cost to move in this direction.

            3. If Move cost < Lowest Cost[neighbor] and **(For Bright Field levels) check for free neighbor grids beside wrong-way wires. Only accept a move if an additional free grid is available (this allows us to make wrong-way wires double width).**

                a. Add the neighbor grid location to the Fronts heap.

                b. Path Trace[neighbor] $\leftarrow$ direction we came from.

g. If path was found to a target:

    i. Retrace from the target back to the source, adding design shapes.

        1. Position ← Target Location

        2. State 0: Initial State

            a. Position ← Path Trace[Position]

            b. State ← 1

        3. State 1: Start of a line segment

            a. If bright field and wire is not running in primary direction: line width ← 2× level width.
            Otherwise line width ← level width.

            b. If dark field: Mark a PSM Blockage beyond the line endpoint.

            c. Starting Point ← Position.

            d. Routing Grid [Position] ← used.

            e. If not primary direction and bright field: Routing Grid[Position's neighbor] ← used.

            f. Owner[Position] ← this net.

            g. Position ← Direction pointed to by Path Trace[Position].

            h. If New Position is in same direction as previous position (still in a line): State ← 2.

            i. Else: State ← 3.

        4. State 2: Point along a line segment.

            a. Routing Grid [Position] ← used.

            b. If not primary direction and bright field: Routing Grid[Position's neighbor] ← used.

            c. If dark field: Mark a PSM Blockage beyond the line endpoint.

            d. Owner[previous point] ← this net.

e. Position ← Direction pointed to by Path Trace[Position].

f. If New Position is in same direction as previous position (still in a line): State ← 2.

g. Else: State ← 3.

5. State 3: End of a line segment.

a. If Starting Point and Current Position are equal, create a rectangle in layout. This is a degenerate point.

b. Else Create a line in the layout: From Starting Point to Current Position with line width.

c. State ← 0. Do not get a new point.

ii. Release the Fronts heap.

The router is implemented with the following assumptions regarding placement of wires:

1. Wires can be placed on adjacent grid points without violating minimum spacing.
2. Wires can end on adjacent grids without violating minimum spacing.
3. Wires can be placed on the grids nearest the boundaries without concern for what wires lie beyond the boundary (there is assumed to be a guard ring of open space at least one grid point in width/height lying just beyond the boundary).
4. Shapes (Bright Field) or Spaces (Dark Field) that have minimum width are Critical.
5. Shapes (Bright Field) or Spaces (Dark Field) that are twice the minimum width (two grids) are NonCritical.

The phase correct router implements four types of layers:

1. No phase restrictions. Wires can be routed horizontally or vertically without restriction.
2. Bright Field phase correct. Wires that are routed in the "wrong way" direction are routed at twice the standard width, and block two adjacent grids.
3. Dark Field phase correct. Wires can only be routed in the primary wiring direction. Each pin is checked for legality (does the pin lie adjacent to another pin or a blockage shape). Each wire end is guarded by one additional grid that is left open to create a space that is twice the minimum space.
4. Dark Field phase correct without additional blocked grids. This is a simplification of layer type 3, and is used to test some assumptions regarding blockage in the switchbox routing test cases.

Test Cases

Three groups of test cases are used. The first set of test cases are hand-drawn sets of nets and pins, used to test the basic functionality of the router and to test the bright-field phase correctness. The results of the hand-drawn test cases are not included.

The second set of test cases are classic routing examples taken from the literature [17]. The glory days of switchbox routing are past us and we were unable to acquire electronic versions of these test cases, so they were entered by hand from diagrams in papers on earlier routers. These test cases required some modification, in that some placed pins on one routing layer on all four sides of the switchbox. This pin arrangement violates the third dark-field wiring restriction. The pins in these test cases were assigned to the layer that is wirable in the direction the pins are aligned. For example, if M1 runs

horizontally and M2 runs vertically, the left and right pins are assigned to M1, and the top and bottom pins are assigned to M2.

The third set of test cases are taken from chip designs that are available to the authors in the form of completed layouts. These test cases are macros from two complete microprocessor designs. For this third group of test cases, a "de-wiring" program was written to find all of the nets in a design, identify all of the shapes in each net, and find the pin locations for each net. The output of the de-wiring program is a list of nets and pin locations, a list of blockages (power grids and other background shapes), and the size of the routing box.

Dark Field Assignment and Checking

Bright Field phase coloring tools are available, and verifying the colorability of the bright field layouts is now relatively straightforward. Two methods of checking Dark Field Phase coloring are described. The first method is to apply known techniques for coloring Bright Field phase shapes, such as those in Berman et al. [7]. The obvious difference between Bright Field and Dark Field is that the Dark Field phase coloring program does not need to create phase shapes. Each unioned (the union of all touching shapes on a level) wire in the layout that has a minimum spacing to another wire, must be assigned a color. A wire's color must be different from the color of every other wire at minimum spacing.

The second method for checking Dark Field phase assumes that the layout is intended to be phase correct. If this assumption is correct, the layout can be checked for correctness by checking for compliance with the routing restrictions. The coloring can then be applied in rows (for a horizontal layer, columns for a vertical layer), and be known to be conflict free. The steps needed to verify the layout are:

- Each shape must be a rectangle.
- Each rectangle must be oriented in the primary wiring direction for the layer it occupies.
- Each shape must be on grid and have the correct width (minimum width for this layer).
- Each shape must have an empty grid point beyond each end.

Performance of Existing Routers

The test cases that were taken from completed chip designs were run through both of the Dark Field checking methods in order to determine "how close" a standard routing solution is to being phase correct. In Table 4, the phase routing errors for the pre-existing routing is broken out into three types of errors. The first type of errors are actual odd-cycles in the phase conflict graph. The second type of errors are violations of the routing restrictions. The third error class is Illegal Pins. These are pins which violate restriction 3. Illegal pins represent unavoidable violations of the phase correct routing restrictions. Illegal pins, in some cases, indicate a methodology issue that needs to be addressed in the definition of how the elements of the hierarchy interact.

Test case P4 is interesting in that it represents a bit stack type topology, where there is a great deal of regularity to the routing. Much of P4 may have been laid out by hand. Any error which occurs once in P4 occurs many times, with some regularity. In P4, many of the pins (nearly 15%) are illegally located, and there are 2495 shapes which contain wrong-way wiring. There are also numerous "comb" shaped figures in P4. The M1 pins trapped within these combs are not routable. These pins lies between vertical M1 blockages (to the left and right), and under vertical M2 blockage. There are 74 pins

on M1 trapped within these combs. In the original layout, these are accessed via bent M1 wires.

In Figure 18, we see a trapped pin on M1. M1 blockage is crosshatched from lower right to upper left, M2 blockage lies over most of the image (cross-hatched from lower left to upper right). The pins are the black squares. An M1 wire accessing these pin cannot travel north/south due to the Dark Field routing restrictions, cannot via directly up to M2 due to blockage, and cannot escape east/west due to M1 blockages, so these pins cannot be accessed.

When browsing a layout which has been routed using traditional routers, it is easy to believe that the routing nearly follows the phase correct routing restrictions. The wires tend to be routed in a dominant wiring direction on each routing plane, on a clearly defined routing grid, with a minimum of jogging from track to track. However, when the wiring results are closely scrutinized, the results are often far from phase correct, regardless of the definition used (odd cycles or phase correct restriction violations).

Test case P4 was very challenging. From the layout, it appears too have been hand-designed as a bit stack macro, making it a poor choice for a switchbox type router.

When these test cases were routed using the phase correct router, there were some phase coloring errors found. All of these coloring errors were the result of illegal pins identified in Table 4. Phase Coloring violations from a traditional router. These represent end-to-end pins on a minimum pitch. If a wire is run alongside the pair on either adjacent track, an odd-cycle coloring error is created.

Phase Correct Routing Results

The tests were performed on an Intel 3.0 GHz system running Redhat 7.2 Linux under VMWARE. The results of the wiring show some very interesting trends with

respect to the classic switchboxes (Table 5). In one case, Thread, the phase correct router achieves better results when obeying the routing restrictions. When routing without restrictions, the use of stair-step routes blocks an excessive number of grids on each wiring layer, and the routing does not complete. In this case, the routing restrictions (primarily the restriction that there be no jogging) provide order to the router. However, in the other switchbox cases, the success rate with routing restrictions is low. This appears to be due to the fact that these switchboxes are very densely filled, and the blocked but unused grid points required by the restrictions leave little room for the final wires to complete routing. In Gerez and Herrmann [17], the provided solutions have as little as 4% of the grid points on a layer available, with an average of around 17%. In the phase correct routing, each wire segment within the switchbox blocks its own length, plus two additional grid points. None of these switchboxes has an extent of more than 23 wiring tracks, meaning that a wire which runs in a track will block approximately 10% of the track with unused and blocked grid points, over and above the actual extent of the track used by the wire. To test the impact of the blocked but unused grid points, the switchbox test cases were routed a second time, using a set of rules which only allows primary direction wiring, but which does not insert an extra blocked wiring space at the end of each wire (Table 6). The Thread test case did not improve. The results for the Pedagogical test case became slightly worse. The other 3 test cases improved substantially when the PSM blockages were not present. The remaining test case results, shown in Table 7, show a small number of missed connections and reasonable runtimes.

Design Impact

Via Counts

It is reasonable to expect that these routing restrictions will cause an increase in via counts, due to the inability to make one or two channel jogs on one wiring layer. It is possible that the wiring lengths will also be impacted. In order to quantify the impact, the router was run on each design using non-phase correct rules, with jogs discouraged but allowed. This allows a direct comparison between the results with and without routing restrictions. The number of vias increased by an average of 16.5%, while net lengths decreased by 5.5%. Larger via count increases were seen in the 2-level wiring cases, smaller increases in the 3-level wiring cases.

Jogs in Phase Correct Layouts

Can a layout be produced which is phase correct, but contains jogs? The answer is a qualified yes. The obvious solution, which would change the phase of the wire at the jog, is prohibited by lithography constraints. Each “as-unioned” shape (i.e. the entire shape consisting of unioned smaller shapes) must carry one phase. However, if a wire were to jog an even number of channels, it could carry the same phase for its entire length, and not produce a phase conflict with adjacent wires (Figure 19). Along the jogged section, the empty grid points for wire ends would prevent phase conflicts.

If a wire were to jog an odd number of channels, it could carry the same phase, if it had an extra empty wiring grid on each side of the ‘wrong phase section’. This restriction would waste considerable space for long wire sections which run in the wrong phase channel. For short wires, this restriction may not prove too disruptive. In Figure 20, the gray sections are wire channels that are forbidden for wires which would normally occupy that channel.

Line End restrictions and Pin Location restrictions

Can a layout be produced which is phase correct, but which ignores line-end or pin restrictions? Here the answer is somewhat more difficult. If there is no empty grid between pins or line-ends, then the phases of the wires and/or pins must differ. This requires two wires in the same channel to have different phases, or one must jog into this channel from an adjacent channel (or an odd number of channels away).

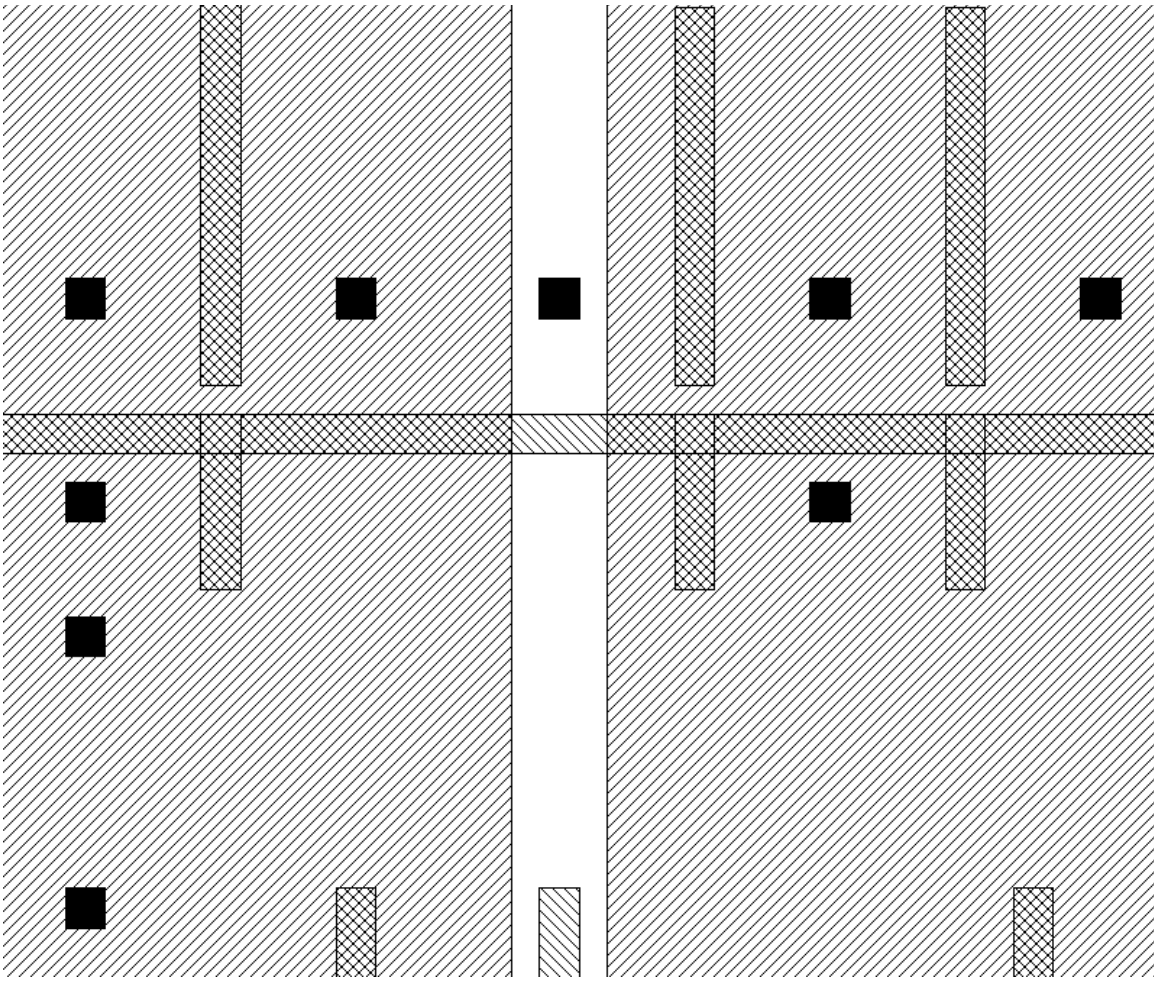

Figure 18. P4 trapped M1 pins between M1 blockages, under M2 blockage.

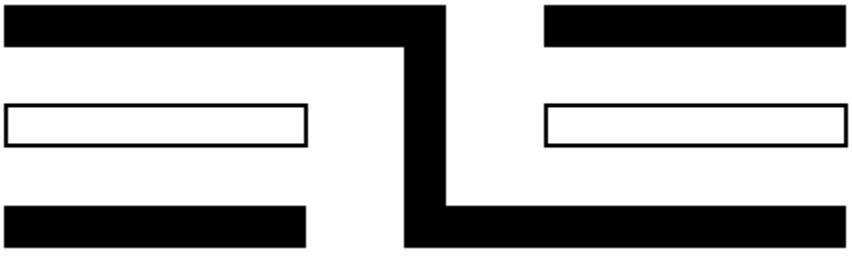

Figure 19. An even length jog.

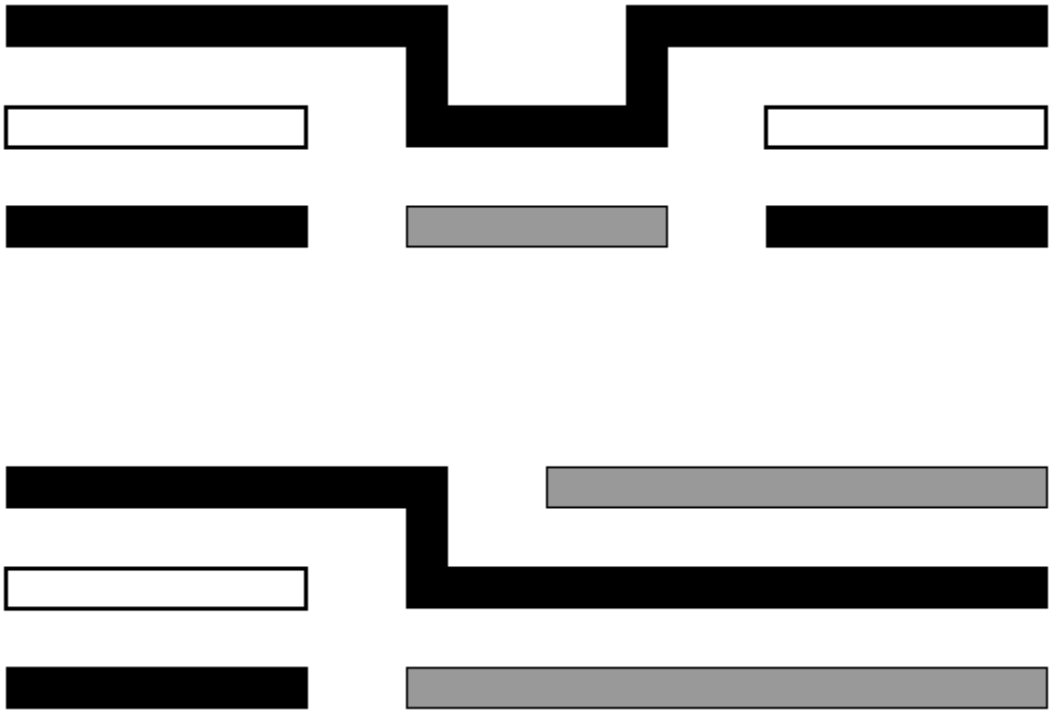

Figure 20. Odd length jogs and the forbidden wire channels they create.

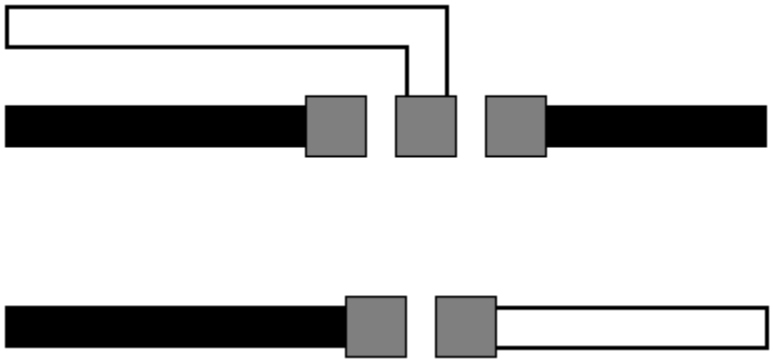

Figure 21. Pin or via colorings for line ends.

Table 2. Literature Test Cases

Name	Nets	Pins	Blocked M1/M2/M3	Levels
Thread	6	19	0%	2
BursteinDifficult	10	24	0%	2
Channel_joo6	10	30	0%	2
Pedagogical	22	57	0%	2
MoreDifficult	25	66	0%	2

Table 3. Microprocessor Macro Test Cases

Name	Nets	Pins	Blocked M1/M2/M3	Levels
P1	1956	5103	17.8% / 8.8%	2
P2	1987	5227	14.7% / 9.4%	2
P3	234	972	16.8% / 13%	2
P4	2892	9811	4% / 12.5%	2
P5	526	1301	16.9% / 3.2% / 11%	3
P6	389	908	15.5% / 2.9% / 10.5%	3
P7	147	351	17.8% / 4.4% / 24.3%	3
P8	130	377	13.9% / 2.3% / 12%	3

Table 4. Phase Coloring violations from a traditional router.

Average Violations	P1-P3	P4	P5-8
M1 Odd Cycles	245.3	786	6.0
M1 Routing Restriction Violations	749.7	4008	9.0
M2 Odd Cycles	107.3	0	22.5
M2 Routing Restriction Violations	157	13	42.0
M3 Odd Cycles	n/a	n/a	0.75
M3 Routing Restriction Violations	n/a	n/a	2.5
Illegal Pins	121.3	1450	46.5

Table 5. Switchbox routing results with AltPSM Blockages.

Test case name	Unrouted Nets	Unrouted Pins
Thread	1	2
BursteinDifficult	3	4
Channel_joo6	2	4
Pedagogical	6	11
MoreDifficult	4	6

Table 6. Switchbox Routing without any AltPSM Constraints.

Test case name	Unrouted Nets	Unrouted Pins	Time
Thread	4	7	0.01s
BursteinDifficult	1	2	0.01s
Channel_joo6	1	1	0.01s
Pedagogical	8	15	0.02s
MoreDifficult	8	18	0.04s

Table 7. Routing results for Phase Correct Routing.

Test case name	Unrouted Nets	Unrouted Pins	Routing Time
P1	8	12	80s
P2	9	12	1048s
P3	6	8	25s
P4	28	40	1298s
P5	0	0	51s
P6	0	0	42s
P7	0	0	11s
P8	0	0	5.1s

CHAPTER 4
TECHNOLOGY MIGRATION

Introduction

In order to maximize designer productivity and meet very aggressive schedules for early test vehicles, it is necessary to migrate layouts from one technology to the next wherever possible. While automatic routers can rewire a layout to phase correctness, there is sometimes considerable hand wiring in a design. Test chips are sometimes designed and submitted to manufacturing before the place and route methodology for a technology is in place. The burden of modifying a wiring layout to meet AltPSM rules then falls to the layout migration tool.

Technology migration has been studied for the migration of layouts to Bright Field AltPSM correctness [23], where the layout must be modified to allow the creation of phase shifter shapes. The challenge for migration is to modify an existing layout to meet the topology requirements of AltPSM.

Technology Migration

Technology migration, or alternatively, physical design migration, is the conversion of a completed, ground rule correct layout from one technology to another technology. Usually migration refers to the conversion of a layout (in shape form) from one technology to a newer, denser technology.

The inputs for a technology migration are a complete layout, a new set of ground rules (the old set of technology ground rules are largely irrelevant), and performance goals (usually in the form of new device sizes).

Technology migration developed as an extension of layout compaction. Layout compaction has the goal of minimizing the area of a layout. A compactor does this by driving all of the shape sizes and spacings to the minimum legal values (typically using a longest-path graph formulation). By compacting with the new set of ground rules, a compactor can convert a layout to the new technology. However, a good migration will preserve the designer's "intent" as well as produce a clean layout.

Intent is hard to define and to measure. One method of defining intent is the relative sizes and spacings between shapes in the layout. In a typical layout, not all shapes are at minimum size or space. It is frequently the case that shapes in a layout will be at non-minimum sizes. Driving the shape sizes to minimum size may impact the performance (by increasing the resistance and capacitance of the wiring paths). Driving the shapes to minimum spacing may cause an increase in wiring noise.

A technology migration is generally considered successful if it is able to:

1. Convert the layout to the new set of ground rules.
2. Introduce no Logical vs Schematic errors (LVS). An LVS error is typically a short (two wires touch that did not touch in the original layout) or an open (a wire in the original layout is disconnected). In essence, an LVS error occurs when the layout's wiring does not match the network that the designer intended.
3. Remove all (or more typically, the vast majority) of the DRC errors caused by applying the new ground rules to the old layout.
4. Preserve the design intent by only making the changes necessary to meet the above goals.

Prior Work

The general migration problem, converting a layout from one set of design rules to another (usually more restrictive and denser) set of design rules has been described in detail in Reinhart [51]. Migrating a layout while minimizing the number of changes is considered in Heng et al. [22]. When a layout is constrained by a fixed boundary or by fixed locations for some objects (such as pins), automatic migration will not typically achieve 100% removal of DRC errors [43,51]. The goal of layout migration is therefore to minimize the manual effort needed to achieve 100% DRC clean results in the target technology.

Migrating a layout from general topologies to BF AltPSM correctness was considered in [23]. That work concentrated on the local problem of removing features that were known to create topologies that cannot be colored.

We consider the problem of migrating routing layers from unrestricted topologies to conservative topologies that can be trivially phase colored.

Migrating a Layout to Phase Correctness

In order to migrate a layout to phase compliance, we identify jogs on the AltPSM layer and move them to another [non-AltPSM] layer. The jog removal process leaves unidirectional wires that can be trivially phase colored. Standard technology migration techniques are then used to legalize the results on the non-AltPSM layer and to separate the wire ends on the AltPSM layer wherever the same color spacing is not met.

We move jogs to another layer in two steps. First, we find all jogs that can be moved into an empty track on another layer. Second, we move the remaining jogs into new tracks inserted into the other layer. The challenges are to move as many jogs as

possible in the first phase, and to add as few new wiring tracks as possible to the layout in the second phase.

In migrating the layout to DF AltPSM compliance, it is necessary to grid the wires and maintain spacings that depend on the colors assigned to the wires. It is no longer sufficient to maintain a minimum spacing between two wires; the required spacing required depends on the phase colors assigned to the wires. We assume wiring rules that require an empty wiring track between all wires of the same color. These spacings apply for both side-to-side and end-to-end relationships.

Definition 1. *Layer 1 is the layer containing the wire that jogs from one channel to another. Layer 1 is being migrated to a DF AltPSM-correct topology*.

Definition 2. *Layer 2 is the wiring layer where we will move the jogged portion of the wire.*

Definition 3. *The jog interval is the range of the wires attached to a jog, in the preferred wiring direction. For a vertical jog, this is the distance between the lowest X and highest X of the horizontal wires attached to the jog. The jog interval is limited by any other wires that share tracks with the non-jogged portion of the wire. See* Figure 22 *and* Figure 23.

Definition 4. *The jog extent is the number of tracks on layer 1 spanned by the jog. For a vertical jog, this is the height of the jog. See* Figure 22.

Definition 5. *A jog cluster is a group of jogs with mutually overlapping intervals and extents. All of the jogs in a jog cluster must be placed taking into account the placements of the other jogs in the cluster. See* Figure 24.

Two jogs interact if their extents and intervals overlap. If two jogs interact, clearly we cannot move the jogs from both wires into the same track on layer 2. If we were to do so, then the jogs would overlap and create an electrical short between the two wires.

Two jogs are coincident if and only if they interact and a wire attached to one jog is in the same horizontal track as a wire attached to the other jog (see Figure 25). Clearly if two jogging wires share a wiring channel in this way, there is a left to right relationship that must be maintained between the jogs in order to avoid creating an electrical short between the wires.

Definition 6. *The jog interaction graph is a graph representation of the legal positions for each jog in a jog cluster.*

A jog interaction graph has one vertex for each jog and an undirected edge for every pair of jogs that interact. The connected components of the jog interaction graph define jog clusters. Isolated jogs form clusters with only one vertex. Note that a cluster may contain both coincident jogs (Figure 25) as well as jogs that are not coincident (Figure 24).

Migration Methodology

Migrating a layout to phase correctness requires 5 steps:

1. Identify the jogs. Find all of the jogs on the AltPSM layer, layer 1. These jogs will be moved from layer 1 to layer 2 by the migration process.
2. Shift jogs to layer 2 one cluster at a time. Shifting a jog means finding an open position on layer 2, within the jog interval, and moving the jog to layer 2 in that new position.
3. For those jogs that cannot be shifted, find the minimum number of new tracks required on layer 2 and insert the jogs illegally into layer 2.

4. Phase color layer 1. Once the jogs on layer 1 have been moved to layer 2, the phase coloring problem on layer 1 is reduced to calculating the parity of the wiring track for each unidirectional wire.
5. Apply technology migration techniques to meet end to end spacings on layer 1 and to spread wires on layer 2 to accommodate jogs inserted between existing wires.

Shifting Size 1 Clusters

An isolated jog can be shifted anywhere along the jog interval to a position where layer 2 is free.

Shifting Larger Clusters

First, two jogs A and B that interact (i.e., have overlapping intervals and extents) cannot be assigned to the same track on layer 2.

Second, coincident jogs A and B (i.e., A and B have ends in the same wiring track on layer 1) must maintain a left to right relationship (if jog A is left of jog B on layer 1, then A must be left of B on layer 2).

Definition 7. *Shifting Clustered Jogs requires finding a maximum number of jogs within a cluster that can be simultaneously moved from layer 1 to layer 2, while not creating any wire collisions between the wires making up the cluster.*

As shown in Figure 26, jogs in a cluster can pass each other to find open tracks on layer 2. In the case shown, jogs A and B can pass each other without creating a short. This differs from the situation shown in Figure 25, where the two jogging wires share a wiring channel and cannot be allowed to pass each other.

ILP formulation for shifting clustered jogs

The problem of assigning a maximum number of jogs in a cluster to new locations can be formulated as an Integer Linear Programming problem (ILP).

The jogs in the cluster are labeled $\{1\ldots N\}$.

The vertical tracks are labeled $\{1\ldots D\}$.

$$x_{jk} = \begin{Bmatrix} 0: \text{ Jog j is not assigned to track k} \\ 1: \text{ Jog j is assigned to track k} \end{Bmatrix}$$

If track k is not a legal assignment for jog j, then $x_{jk} = 0$, otherwise $x_{jk} = \{0,1\}$.

$\forall j, \sum_{k=1..D} x_{jk} \le 1$: Every jog j is assigned to at most one track.

For every pair (i,j) of interacting jogs, we add the constraint $x_{ik} + x_{jk} < 2$: Jogs I and j cannot both occupy track k.

$x_j = \sum_{k=1..D} k x_{jk}$: Sets x_j to the track assigned to jog j.

For every pair (i,j) of coincident jogs, we add the constraint $x_i < x_j$: Jog I must be to the left of jog j (as shown in Figure 25).

Any assignment of $\{0,1\}$ values to the x_{jk} variables that satisfies these constraints, defines a feasible allocation of jogs to layer 2.

To maximize the number of jogs assigned to layer 2, we include the objective function: maximize: $\sum_{j \in N, k \in D} x_{jk}$

Greedy heuristic implementation for shifting clustered jogs

A greedy heuristic for assigning clustered jogs has also been developed. The heuristic is based on the construction of a "conflict graph", G.

Definition 8. *A conflict graph G for a jog cluster is a graph that contains a node for every legal jog location. There is an arc connecting every pair of jog locations that cannot be occupied simultaneously.*

The Conflict Graph G is constructed as follows:

1. For each jog, create one node for every open position in layer 2 within the jog's interval. These nodes are N_{i1} … N_{im} for jog i.
2. If jog i and jog j cannot occupy position y without a collision (y is open for i and j, and the intervals and extents for jogs i and j overlap), then insert an arc between N_{iy} and N_{jy}.
3. If jog i must be to the left of jog j, insert an arc between all nodes N_{iy} and N_{jw} where $y < w$.

Construction of the initial Conflict Graph takes time $O(N^2)$, where N is the number of jogs in the cluster.

In Figure 27, we see a pair of jogs A and B, and the conflict graph that is created from these jogs. Step 2 of the conflict graph creation inserts arcs wherever two jogs will overlap and create a wire short.

In Figure 28, we see a pair of jogs, A and B, with a left to right constraint caused by a shared wiring channel on layer 1. In this case step 3 of the conflict graph creation algorithm inserts arcs from each node for jog A to all of the nodes for jog B that are below or to the left.

The greedy heuristic for shifting clustered jogs using G is as follows:

1. Sort the nodes in G by increasing degree.
2. Select the lowest degree node in G: N_{ij}

3. Assign jog i to location j.
4. Remove all arcs connected to N_{ij} and the nodes at the other end of these arcs.
5. Remove Nodes $N_{i1}…N_{im}$ from G.
    a. Remove all arcs connected to these nodes from G.
6. If unassigned jogs remain, return to step 1.

For N jogs with M layer 2 tracks, there are O(NM) nodes and $O(N^2M^2)$ arcs. If the jog shifting heuristic is implemented using a Fibonacci heap, its complexity is $O(N^2M^2)$.

Jog Insertions

Once the trivial jogs and clustered jogs have been moved to layer 2, the remaining jogs are those that have no layer 2 opening within their extent or that have conflicts within a cluster that prevent them from reaching an open layer 2 track. These jogs will require additional tracks to be added to layer 2. Adding tracks to layer 2 is done by inserting these jogs between existing layer 2 wires. Since the existing layer 2 wires are at minimum spacing already, we are creating a legalization problem for the migration step that follows (spreading the layer 2 wires to make space). For example, in Figure 30, we move a jog onto layer 2, and then spread the existing layer 2 wires to accommodate the moved jog.

We want to place each jog into a location within its jog interval while minimizing the number of additional tracks added. Given two jogs with overlapping intervals, we could insert each independently, thus adding two layer 2 tracks. However, if we can insert them into layer 2 in the same track (provided they do not have overlapping extents), then we are only adding one additional track. In Figure 29, we see two alternatives for inserting two jogs. If we inset them independently, as shown in the top

example, we insert two new wiring tracks into the design. If we insert them into a common channel, we insert only one new wiring channel into the design.

The jog insertion problem maps well to the minimum set cover problem, and from that problem we find an ILP implementation, a greedy heuristic, and a reduction.

Definition 9. *The Jog Insertion problem requires moving a set of jogs from layer 1 to layer 2, inserting layer 2 tracks to hold the moved jogs, while minimizing the number of tracks added to layer 2.*

Jog insertion is NP-hard

We show that jog insertion is NP-hard by reduction from Minimum Set Cover [16].

Inputs:

Set Elements: $U = \{1 \ldots N\}$.

Subsets of U : $T = \{T_1 \ldots T_M\}$.

Problem:

Find the smallest $S \subset T$ such that $\bigcup_{i=1}^{|S|} S_i = U$.

Formulation:

1. Define a wiring box 2N channels high (0..2N+1) by M+1 wiring channels wide (1…M+1). There are two wiring layers, L1 and L2. All channels on L1 and L2 are initially open.
2. Define wires on L1: $W(i,t_i), 0 \le i \le N-1$. Each $W(i,t_i)$ spans the wiring box from left to right, beginning on horizontal track 2i and jogging to track 2i+1 at an arbitrary vertical track. Each wire corresponds to one set element.

3. $\forall W(i,t_i), \forall t_i \not\subset T$ insert blockage into the region: (T,2i) to (T+1,2i+1) on L2. This blocks the channels on L2 that correspond to all subsets of U that do not contain element i. In the jog assignment problem, these blockages correspond to regions where wires cannot be inserted (such as blockages caused by power buses or other reserved regions).
4. Define jog intervals for each of the wires in step 2.
5. Find the minimum jog assignment for the jog intervals in step 4. This will be the minimum set of vertical channels necessary to intersect all of the jog intervals.
6. The minimum set cover for U will consist of the subsets corresponding to the channels chosen by jog assignment.

Each Jog Assignment exactly matches one Set Cover, establishing a 1:1 correspondence between the Jog Assignment and the Set Cover.

In the Jog Assignment created from the Set Cover, all of the layer 2 wiring tracks are occupied, and a selected set of the positions between wires are unavailable.

In Figure 31, we have an example Jog Assignment problem built from a Set Cover problem. In this Set Cover instance, there are 4 set elements {1..4} and 5 subsets {A..E}. Set element 1 is included in subsets B and E, element 2 in E, clement 3 in subsets A and D, and element 4 is included in subsets B and D. The set elements correspond to wires on layer 1. The subsets correspond to the positions between layer 2 wiring tracks (the layer 2 wiring tracks are shown as vertical lines in the figures).

The minimum Jog Assignment and Set Cover for this problem selects subsets D and E as shown in Figure 11.

ILP formulation for jog insertion

The problem of finding a minimum number of tracks to insert can be formulated as an Integer Linear Programming problem.

The jogs and their intervals in the layout are labeled $\{1 \ldots N\}$.

Constants: $I_{ij} : \begin{Bmatrix} 0\text{: Jog j cannot be located in interval i.} \\ 1\text{: Jog j can be located in interval i.} \end{Bmatrix}$

$X_i = \begin{Bmatrix} 0\text{: Interval i not chosen} \\ 1\text{: Interval i chosen} \end{Bmatrix}$

$\forall j, \sum_{i=1 \ldots N} I_{ij} X_i \geq 1$: Every jog j must be located in at least one chosen interval.

To minimize the number of intervals chosen, and number of tracks inserted, we include the objective function:

$$\min : \sum_{i=1 \ldots N} X_i$$

Greedy heuristic for jog insertion

The greedy heuristic for Jog Insertion works by finding the layer 2 track that intersects the most intervals, selecting that location for jog insertion and repeating. This greedy heuristic is a standard method for minimum set cover [56].

1. Sort the jog intervals by starting point (lower coordinate).
2. Scan the layout from left to right (or top to bottom depending on the predominate wiring direction). Keep a running total of the number of intervals currently active, and keep the high water mark (maximum number of intervals active at any track position).
3. At the high water mark, take all of the jogs with intervals intersecting that position. Assign all of these jogs to this track.

4. Remove the assigned jogs and their intervals from the problem.
5. Repeat from step 2 until no jogs remain to be assigned.

For N jogs, the complexity of this operation is $O(N^2)$.

Results

The ILP and greedy heuristics described have been implemented and run on a set of macros taken from several microprocessor designs. The tests were performed on an Intel 3.0 GHz system running Redhat 7.2 Linux under VMWARE.

For each macro, the jogs were removed, the wires colored on layer 1, and the layout migrated with a set of ground rules that enforced DF AltPSM spacings between same color wires. The key indicators of success are the number of jogs that can be moved without inserting tracks, the number of tracks inserted, and the performance of the heuristics compared with the ILP implementations (both in success rates and in runtime). In Table 8 we give the characteristics of the test cases used to verify the performance of these algorithms. The jogs break out into three groups, trivially moved jogs, clustered jogs, and jogs inserted illegally between existing wires. The "tracks" count is the number of wiring tracks in the direction that jogs are being inserted into layer 2. If jogs are vertical, they are inserted into vertical wiring tracks, and the number of vertical channels in the test case is counted as "tracks."

The cluster sizes found in the layouts range from 2 jogs up to 82 jogs, with the majority of clusters containing 2 to 5 jogs. The jog extents range in size up to 62 tracks. The most frequently occurring jog was a 1 track jog. The majority of the jogs were less than 10 tracks in size.

compares the heuristic with the ILP formulations for both of the methods used to move jogs to layer 2. The clustering heuristic always succeeded in placing as many jogs

as the ILP method, but did so considerably faster. However, the average sizes of the jog clusters is such that the runtimes for the ILP never become unmanageable (recall that each cluster is solved as a separate ILP problem).

The jog insertion heuristic does not produce as good a result as the ILP implementation. The runtimes for both are essentially the same (zero runtime to within 1/100 sec). In numbers of tracks inserted, the heuristic inserts up to 24% more tracks.

Table 10 shows the results of running technology migration after jogs are removed. The initial errors are DRC and phase coloring errors present after the jog removal step. The migration tool is unable to fix all of the errors in the layout when the pins and boundary are held fixed. On layer 1, holding the pin locations fixed causes coloring errors that cannot be removed without manual intervention (the designer must change the pin locations in the lower level circuits). These coloring errors appear as end to end spacings where two pins are in the same wiring track but at the minimum spacing. When inserting jogs between existing wires on layer 2, fixed pin locations limit the ability of the migration tool to bend and slide wires around the newly inserted jog, causing more pin related errors. Migrating the layouts without pin and boundaries fixed results in a very small number of remaining errors (in most cases, zero errors).

Migrating the layouts with the boundary fixed, but the pin locations free to move, results in very few errors. The majority of the errors that remain after jog removal and migration are the result of fixed pin locations.

Migrating the layouts with the boundary and pins free to move results in a negligible number of wiring tracks added, compared with the size of the wiring box and the number of jogs inserted into the layout. Even though some of the layouts appear to be

quite densely routed, there is free space available to absorb the newly inserted jogs, particularly if we insert bends into layer 2 wires to allow them to wrap around inserted jogs.

Removing jogs by moving wires to another layer adds vias to the layout. However, the total number of vias added is small and strictly limited to two vias per jog.

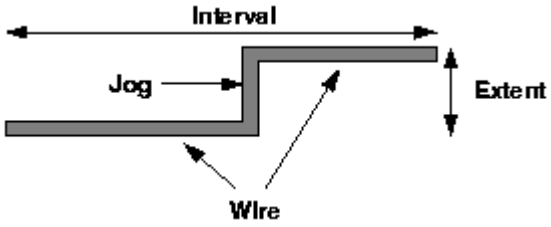

Figure 22. Jog Interval and Extent

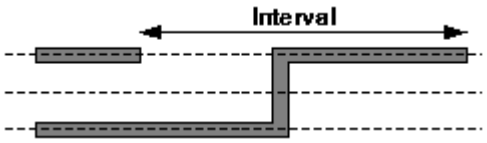

Figure 23. Jog Interval shortened by a wire.

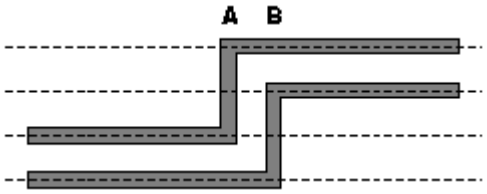

Figure 24. Jogs with overlapping intervals and extents.

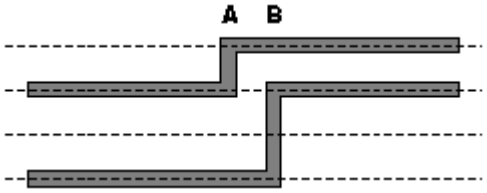

Figure 25. Coincident jogs with a left-to-right constraint.

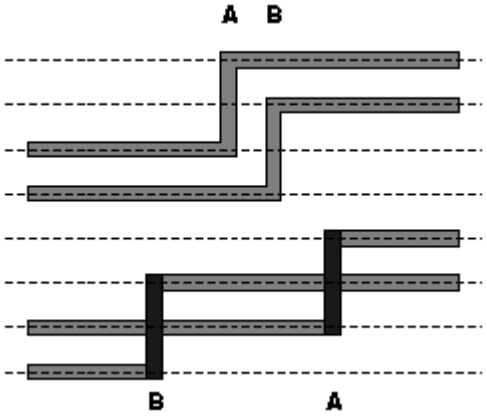

Figure 26. Jogs in a cluster, shifted to layer 2.

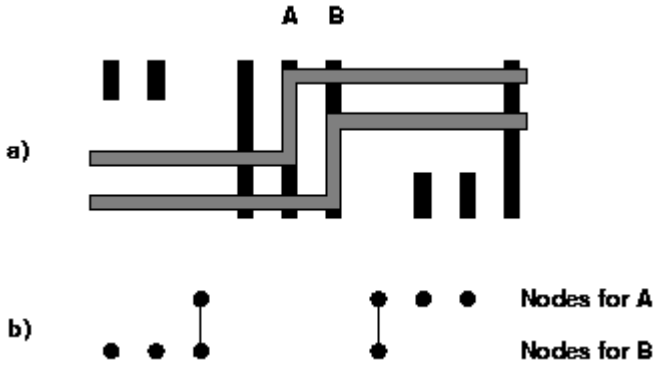

Figure 27. Jogs and conflict graph created from them.

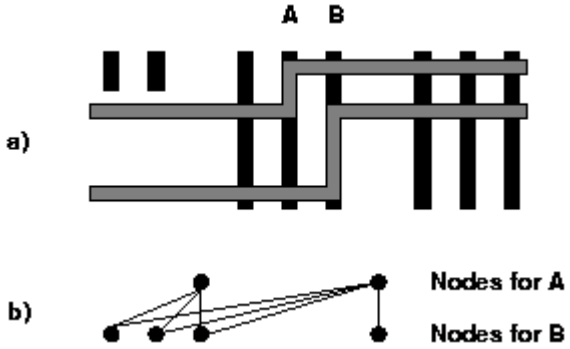

Figure 28. Conflict graph for coincident jogs.

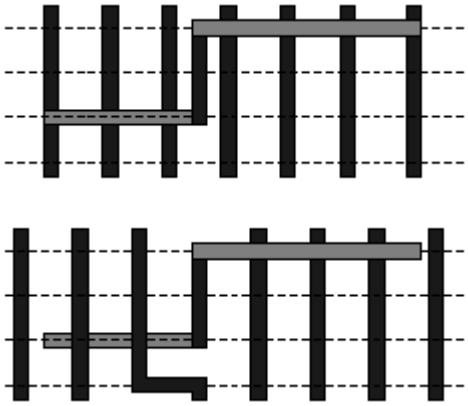

Figure 29. Inserting a jog between two existing wires.

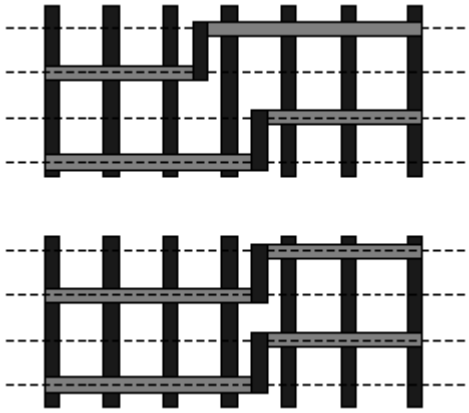

Figure 30. Alternatives for inserting jogs.

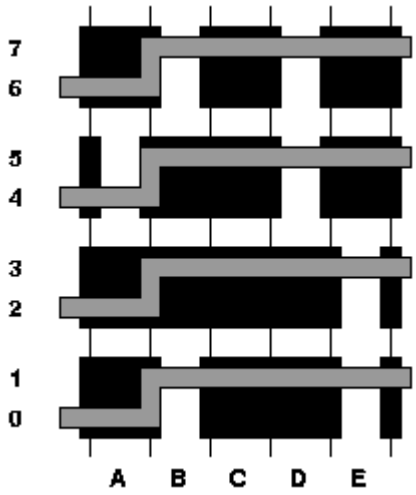

Figure 31. Set cover example.

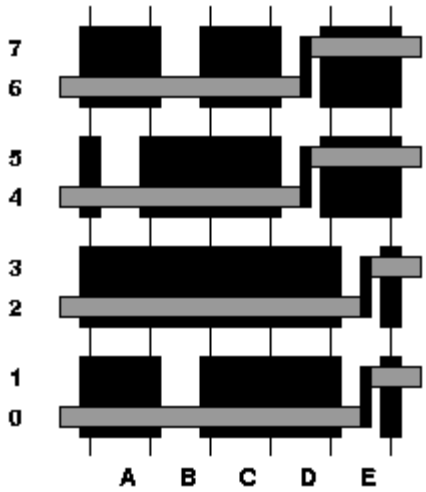

Figure 32. Jog assignment solution to set cover.

Table 8. Jog Removal Test Cases

Macro	Nets	Jogs	Trivially Moved	Clusters (Jogs)	Inserted Jogs	Initial Tracks
A	535	168	58	31 (105)	5	496
B	590	123	51	25 (63)	9	270
C	1136	247	107	48 (107)	33	283
D	5026	547	173	99 (336)	38	1390
E	2450	371	118	75 (233)	22	660
F	1879	217	135	31 (62)	20	348
G	2076	550	194	118 (284)	72	509
H	2098	478	226	98 (226)	67	393
I	2545	275	104	53 (170)	1	746

Table 9. Jog Removal Results

Macro	Cluster Heuristic		Cluster ILP		Jog Insertion Heuristic		Jog Insertion ILP	
	Jogs	Time	Jogs	Time	Tracks Added	Time	Tracks Added	Time
A	105	0.02s	105	0.27s	5	~0.00s	4 (-20%)	~0.00s
B	63	~0.00s	63	0.14s	7	~0.00s	7	~0.00s
C	107	0.01s	107	0.18s	25	0.01s	23 (-8%)	~0.00s
D	336	0.08s	336	1.82s	35	0.01s	30 (-14%)	~0.00s
E	233	0.12s	233	2.81s	22	0.01s	17 (-23%)	~0.00s
F	62	0.01s	62	0.23s	18	~0.00s	16 (-11%)	0.01s
G	284	0.01s	284	14.31s	72	0.02s	57 (-21%)	0.03s
H	226	0.07s	226	0.58s	59	0.01s	45 (-24%)	0.02s
I	170	0.09s	170	1.14s	1	0.01s	1	0.01s

Table 10. Post Jog Insertion Migration Results

Macro	Initial Errors	Pins/Boundary Free		Pins Fixed Errors	Boundary Fixed Errors	Pins/Boundary Fixed Errors
		Errors	Tracks Added			
A	327	0	0	113	0	113
B	308	0	0	120	0	120
C	664	0	0	314	4	314
D	957	0	9	364	3	364
E	484	0	0	75	0	75
F	347	0	0	87	0	87
G	1659	2	1	701	2	701
H	1496	5	0	653	6	655
I	330	0	0	58	0	58

CHAPTER 5
DESIGN FOR MANUFACTURABILITY

Introduction

Design for Manufacturability (DFM) encompasses a number of techniques that target improved manufacturing yields. One of these techniques is Redundant Via Insertion (RVI). Insertion of redundant vias improves yields by attempting to add a second via at every inter-layer connection that only has a single via. The addition of a second via lowers the probability that a single via failure will result in a disconnected wire, improving yield and reliability [21].

By their nature, redundant via programs add jogs onto wires. Each via connects two adjacent layers, and these two layers will typically have orthogonal wiring directions, generally one will be vertical and the other horizontal. It is possible for these two layers to meet at one point, a single via. The addition of a second via requires one of the layers to jog. A strictly gridded layer cannot accept the jog that is necessary to add a redundant via. We examined techniques for adding redundant vias when only one layer is allowed to accept jogs, and quantify the success of various techniques under these topology limitations.

Prior Work

Some previous redundant via insertion tools [2-4] perform a greedy search around the location of existing vias, placing redundant vias into open locations.

More recent work [63] gave a theoretical model for redundant via insertion, by identifying bipartite matching as a model for the problem, but no details are given.

Details of a bipartite matching model are provided by Lee and Wang [34]. They describe a bipartite graph in which every open redundant via location is represented as a node in one set of vertices, and every existing via is represented as a node in a second set of vertices. Each existing via is connected by an arc to each redundant via location that is both neighboring and available. The authors then provide a counter-example for bipartite matching in the form of a stacked via (two vias connecting three layers, located at the same location, with one stacked above the other). The authors then formulate the redundant via insertion problem as a Maximum Independent Set problem and provide a heuristic solution.

A different approach is taken in Allen at al. [5]. They describe the use of a technology migration tool, or "minimum perturbation compactor" to insert redundant vias. This method inserts over each existing via a target shape. The minimum legal size of the target shape is large enough for a second via (and the required space) to be added. A constraint graph is then built from this layout and the layout optimizer is run to create the required space. The failing target shapes are removed and the optimization is run a second time. The target shapes that have grown sufficiently to receive a redundant via are replaced with a via pair.

Layout Restrictions and DFM

Gridded and Alternating Phase Shift layout styles require a greater degree of regularity than unrestricted layouts. Ideally DFM techniques are applied to the design during the design stage. Yield aware routers attempt to insert redundant vias during the routing stage [60]. However, the optimum parameters for DFM may not be known until after the layout is substantially complete, and layouts being migrated from earlier technologies will typically not be designed in a way that follows the newer technology's

DFM guidelines. Post-layout optimizations for DFM, such as RVI and Wire Spreading, tend to increase the complexity of the layout by adding vertices and edges to the existing shapes in the layout. Restricted layout topology design styles generally call for the layout complexity to be reduced, with fewer vertices and edges. This can put the goals of restricted topologies and DFM in conflict.

As an attempt to satisfy both goals a redundant via pair could be placed centered on the wire on one level, so that the amount of off-direction wire is minimized. However, this solution is more disruptive to routing than leaving the original via on-track and placing the redundant via in an adjacent track. As shown in Figure 33, a centered redundant via pair (on the left) requires two additional open wiring tracks, versus one additional track for a pair offset to one side of the original wire (on the right).

There may be some intrinsic conflicts between the goals of DFM tools and some forms of restricted topology layout. For example, the application of wire spreading to a technology with forbidden pitch limitations [55] would potentially be very difficult. In a technology with forbidden pitches, the wire spreading program would need to balance yield improvement and space utilization while restricting wire movement to the available wire spacings. Applying wire spreading while limiting the movement of wires to one entire wiring pitch could significantly limit the gains achieved.

Static Redundant Via Insertion

Static Redundant Via Insertion (SRVI) is the technique of inserting redundant vias without moving or displacing existing wires. The locations available to insert new redundant vias are exactly those locations that are open in the starting layout.

For a two layer problem without stacked via concerns, we can apply the maximum bipartite matching described in Yao et al. [63]. By restricting the problem to a 1-D

problem, inserting vias only in the primary wiring direction, the number of potential conflicts between vias is reduced to two. Each potential redundant via location can only have two existing vias competing for that location. The conflict graphs shown in Lee and Wang [34] become acyclic, with all nodes having degree zero, one, or two. This restricted bipartite matching can be performed optimally in linear time.

In Figure 34, we show a 1-D static redundant via insertion problem, and the conflict graph created. Redundant vias can be placed at locations R1, R2, R3, and R4. Vias V1, V2, and V3 require redundant vias from these four locations. One matching is R1-V1, R2-V2, and R3-V3.

Dynamic Redundant Via Insertion

In Dynamic Redundant Via Insertion (DRVI) we allow existing wires to bend or move out of the way in order to add vias. In order to do this we apply the techniques used in wire spreading [9] to bend existing wires and create additional space for more vias to be added to neighboring wires.

The method we use is based on Allen at al. [5]. Each existing via is covered by a target shape (or marker). The minimum legal size for the target shapes is made large enough for two vias and the minimum via to via spacing. The target shape is restricted to remain within the metal layers above and below the via. After the target shapes for via insertion are placed into the layout, the layout is compacted. Unless there is sufficient space to insert a second via for every existing via, the constraint graph created will contain positive weight cycles.

The problem of deleting a minimum number of arcs from a compaction constraint graph to remove the positive weight cycles was shown in Dong [13] to be NP-hard. However, removing the positive weight cycles by reducing the weights on edges can be

performed using a linear programming relaxation. In this method, a selected set of the constraint values are reduced rather than being removed from the constraint set [13,22]. Using the technique for constraint relaxation described in Heng et al. [22] we optimize the layout using linear programming, In this formulation, each unsatisfied constraint is transformed into a relaxed form. Heng et al. [22] begins with a formulation of the compaction problem as:

Minimize: $\left\| X - X^{Old} \right\|$

Subject to: $X_j - X_i \geq L_{ij} \quad \forall A_{ij} \in A$

The X values represent locations of edges in the layout. X_i^{Old} is the original position of edge i, X_i is the final position of edge i.

This formulation is the minimum perturbation compaction problem. The total movement of the edges in the layout is minimized while finding a solution where every arc (A_{ij}) in the constraint graph satisfies a minimum distance constraint (L_{ij}).

The authors then linearize the problem by adding a weight for every edge (W), and a left (L) and right (R) constraint for every edge in the layout (represented as a vertex in the constraint graph). The left and right constraints represent lower and upper bounds on the location of each edge. The difference between two edges is still constrained by the minimum distance constraint (L_{ij}). Each edge is initially located at the initial position of X^{Old}. The linearized problem is then:

Minimize: $\sum_{V_i \in V} W_i \left(R_i - L_i \right)$

Subject to: $X_j - X_i \geq L_{ij} \quad \forall A_{ij} \in A$

$L_i \leq X_i, L_i \leq X_i^{Old} \quad \forall V_i \in V$

$$R_i \geq X_i, R_i \geq X_i^{Old} \quad \forall V_i \in V$$

The remaining problem is how to solve the linearized formulation when there are positive weight cycles present in the constraint graph. To handle positive weight cycles, the authors describe the following relaxation of the problem definition:

For each unmet constraint arc A_{ij}:

Define new variables: $D_{ij} = X_j^{Old} - X_i^{Old}$

M_i

Set the initial value of: $M_i^{Old} = X_i^{Old}$

Add new constraints: $X_j - M_i \geq D_{ij}$

$M_i - X_i \geq 0$

$M_i - X_j \geq -L_{ij}$

The objective function minimized is then:

$$\sum_{V_i \in V} W_i \left(R_i - L_i\right) + \lambda \sum_{A_{ij}} \left(M_i - X_j + L_{ij}\right)$$

Where $\lambda > \sum_{v_i \in V} W_i$

In this formulation, the location of M_i lies between X_i and X_j. M_i is initially set to the old location of X_i, and is relaxed away from X_j until the distance from M_i to X_j reaches L_{ij} (the unsatisfied constraint value). Upon completion, X_i is set to the value of M_i.

The left hand portion of the objective function is the linearized minimum perturbation layout compaction problem. The right hand portion is the relaxation of the constraints that are not met in the original layout. The value of λ prevents already met

constraints from being violated to meet the initially unmet constraints. No new ground rule violations are added to the layout while fixing the existing violations.

Because the input constraints describe a unimodular problem [50], we do not need to use ILP to solve the problem. The integer (layout grid coordinates) input to the problem results in an integer output solution.

Initially our layout is legal, there are no unmet constraints in the constraint graph. The addition of the target shapes for redundant vias introduces unmet constraints, and it is these constraints that will either be met (we can insert a via at that location) or relaxed to remove a positive weight cycle (we cannot insert a via at that location). We know that in the worst case, it will be necessary to reduce every target shape's size constraint to the size of a single via (if there is room for no redundant vias).

In order to create more available locations for vias to be inserted, we add the ability for each wire on the second layer to bend in order to add vias, as shown on the bottom wire in Figure 35. Using a technique from wire spreading, each wire is broken into a series of collinear segments. The distance between segment insertions is a key parameter in the results.

Each wire is transformed from two nodes in the constraint graph (a top edge and a bottom edge) into two nodes per segment. When the separation between jog insertions is too small, large numbers of added segments are created, dramatically increasing memory requirements and runtimes. When the separation between jogs is too large, the optimizer does not have sufficient flexibility in inserting vias because longer segments of wire must move together.

Insertion of the jogs adds significantly to the number of nodes and constraints handled by the layout optimization tool. The insertion of three jogs into two wires creates an additional 7 constraints. Constraints are required between the additional segments (the vertical arrows in Figure 36) and between the corners created at the jogs (the diagonal lines).

The insertion of jogs allows for the creation of redundant vias in locations that wouldn't be allowed otherwise. In Figure 37, we have inserted a via on M1 while disallowing jogs on M1. The M2 wires have bent out of the way to allow a jog on M2 to accept the additional via.

Results

The SRVI and DRVI techniques were both implemented and run on a set of custom and random logic macros from microprocessor layouts. For the initial set of tests, the jog interval (distance between inserted jogs) was set at twice the wiring pitch. A second set of tests were run varying the jog interval.

For a set of 17 test cases, 1-D SRVI doubled an average of 37% of the vias while 1-D DRVI doubled an average of 67% of the vias. The range for static insertion is 5% to 93%. The range for dynamic insertion is 37% to 99%. The full results are shown in Table 11.

The most remarkable feature of the two techniques is the differences in runtimes. SRVI runs in time that is linear with respect to the number of vias. DRVI requires the construction and solving of a full compaction constraint graph. SRVI ran in an average of 4 seconds. DRVI had an average runtime of 11 minutes, 10 seconds, and increase of 16,650%.

The runtime increase is daunting, but for a post-processing operation that is run once on a macro or chip, and which is expected to measurably increase the yield and reliability of the chip, the doubling of the success is worth the time. It is also worth noting that the runtimes for DRVI are not unacceptably long, but rather the runtimes for SRVI are extremely short, leading to the very large runtime differences.

The distance between the jogs inserted into wires is a parameter that requires tuning. The runtime and memory requirements of the dynamic technique are heavily dependent on the number of jogs inserted into wires. For example, test case A has 173,767 jogs added to wires if the distance between jogs is one-half the wiring pitch. At eight times the wiring pitch, only 5,847 jogs are added. This is substantially fewer than one-sixteenth the number of jogs for one-half pitch intervals, because many wires are shorter than eight wiring pitches, and do not receive any additional jogs.

Via insertion results with varying jog intervals are shown in Table 12. Some models have a decrease in insertion success of 18-28% as the insertion interval is changed from one-half pitch to eight pitches. Adding no jogs to the layout cuts the success rate by as much as 17% (compared with the eight pitch results). Other models are essentially insensitive to the jog insertion interval.

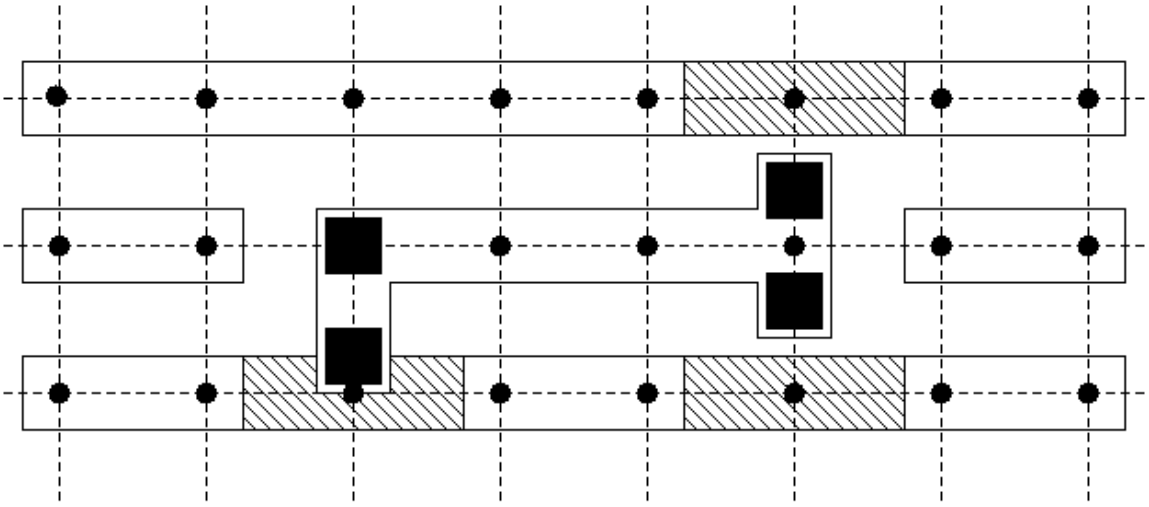

Figure 33. Centered Redundant Via versus Offset Redundant Via

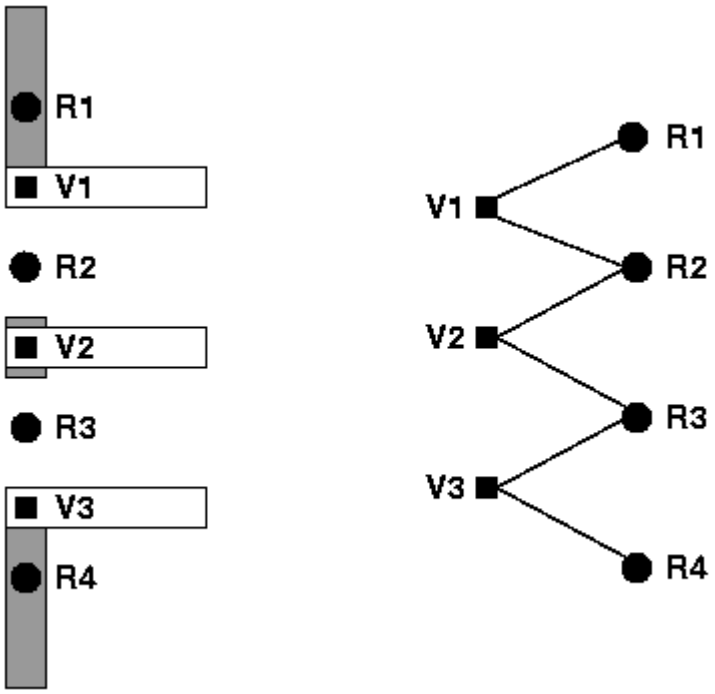

Figure 34. Static 1-D Redundant Via Insertion

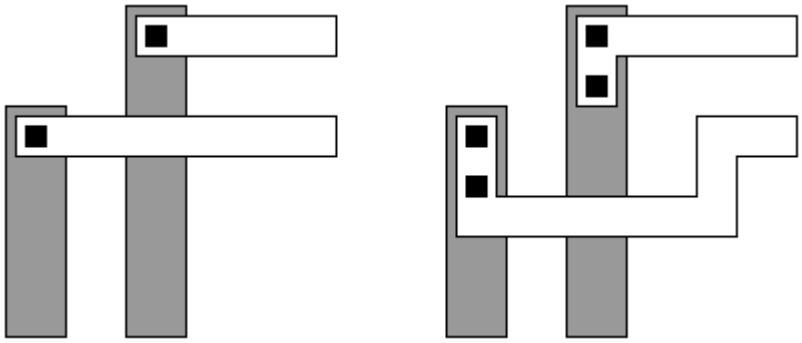

Figure 35. Wires bending to accommodate redundant vias.

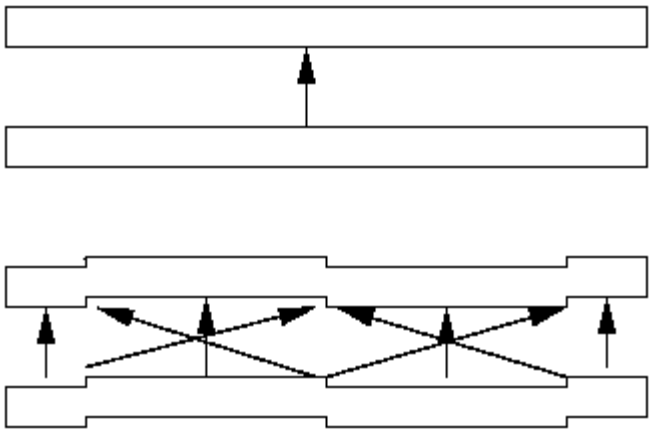

Figure 36. Jogs inserted and additional constraints created.

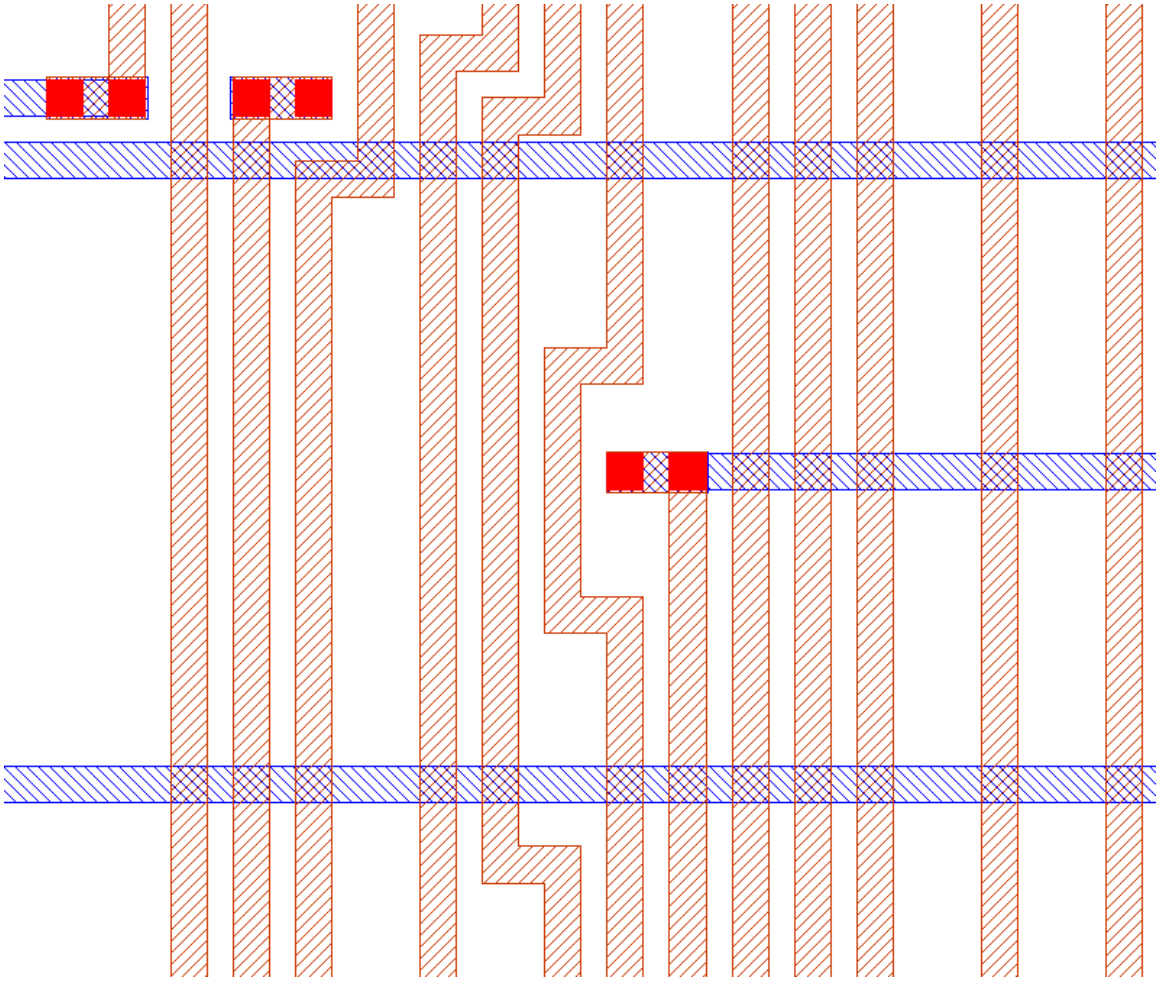

Figure 37. Example of wire bending with via insertion.

Table 11. Redundant Via Insertion Results

Model	Initial Vias	SRVI		DRVI		Via Difference	Runtime Difference
		Doubled	Time	Doubled	Time		
A	7595	68%	3.6s	88%	3m 42s	20%	6,067%
B	8360	65%	4.2s	84%	4m 20s	19%	6,090%
C	7168	68%	3.1s	89%	3m 26s	21%	6,545%
D	4819	62%	2.1s	87%	2m 32s	25%	7,138%
E	1341	93%	1.1s	99%	17.2s	6%	1,464%
F	14262	26%	6.1s	79%	6m 1s	53%	5,818%
G	2942	23%	1.5s	52%	1m 32s	29%	6,033%
H	738	9%	1.0s	88%	12.9s	79%	1,191%
I	2990	15%	1.4s	41%	1m 46s	26%	7,471%
J	4525	8%	1.7s	37%	3m 16s	28%	11,429%
K	19113	34%	9.0s	38%	33m 38s	4%	22,322%
L	3588	17%	1.8s	76%	1m 24s	59%	4,567%
M	22273	34%	12.9s	42%	45m 5s	8%	20,869%
N	1944	11%	1.1s	66%	1m 10s	55%	6,263%
O	22270	5%	8.3s	55%	39m 40s	50%	28,574%
P	26803	6%	9.2s	46%	41m 18s	40%	26,834%
Q	1777	82%	1.8s	98%	29.11s	16%	1,517%

Table 12. Success rate for DRVI with varying jog insertion intervals.

Model	½ Wiring Pitch	1X Wiring Pitch	2X Wiring Pitch	4X Wiring Pitch	8X Wiring Pitch	No Jogs Added
1	95%	93%	88%	82%	77%	60%
2	91%	89%	84%	77%	67%	59%
3	94%	91%	87%	79%	66%	57%
4	99%	99%	99%	99%	98%	97%
5	42%	42%	41%	41%	40%	39%
6	36%	36%	36%	36%	36%	36%
7	76%	76%	76%	75%	75%	73%
8	98%	98%	98%	96%	92%	85%

CHAPTER 6
CONCLUSIONS

Routing

We have shown that it is practical to apply standard routing algorithms, with modifications, and create a correct-by-construction phase shiftable layout. In small, contrived switchbox routing test cases, there is a substantial penalty for leaving an empty grid point between wires in one wiring track. However, in macro layouts the impact of the empty grid points is not significant enough to influence the overall results.

Technology Migration

This work shows that it is possible to migrate a layout to phase compliance, but several key challenges remain:

- Manual work will be required because of fixed constraints, primarily pin locations on lower level circuits. As seen in Chapter 2, fixed pin locations that cannot be legally phase colored must be changed in order to color the layout.
- The need for a phase aware redundant via insertion program, to minimize the yield and reliability impact of adding additional vias to the layout.

Technology changes that require layout topology changes threaten to create discontinuities where layouts from previous technologies cannot be successfully migrated into the newer technology. However, it is possible to migrate layouts across these technology boundaries.

Design for Manufacturability

We examined the results of adding redundant vias with restrictions on the changes made to the layout topology. We have shown that it is possible to add redundant vias to a restricted topology layout with high success rates. We compared a static algorithm, one that does not move wires, to a dynamic algorithm that is capable of shifting wires to create space for additional vias. The static 1-D algorithm runs extremely fast, but did not reach the success rates for via insertion seen in some tools. The dynamic 1-D algorithm was able to reach via insertion rates that match those seen in the literature, but at the expense of much longer runtimes than the static algorithm.

We also quantified the impact of adding jogs to existing wires to create space for redundant vias. These results are highly topology and density dependent. For some models there is no measurable advantage to adding wire jogs. For other models, aggressive wire jogging allows the addition of 35% more redundant vias.

Future Research

Alternatives to Layout Restrictions

There are alternatives to routing restrictions for AltPSM correct routing. The first, and most obvious, is to change the layout to phase correct after routing completion. One approach is to attempt to migrate the layout to phase correctness by applying the routing restrictions enumerated. We show in this work that migrations of this sort are possible. However, if given a blank slate this is rather self-defeating. We would wire without respect to the restrictions, and then attempt to apply the restrictions retroactively. In addition, routers are now being extended to understand timing, noise, and yield considerations. A program which is applied to the completed wiring, to fix phase coloring errors, should ideally understand all of these issues as well (or nearly as well) as

the original router. An algorithm that fixes phase errors, but inserts other design errors will lead to more iteration in the design process. While it may be undesirable, we can wire the design without restrictions, or with some subset of the restrictions, and then attempt to clean up the phase conflicts which are found.

Incremental Coloring

For routing, another possibility is to track the coloring of wires dynamically, and use a net union (combining shapes that touch or have the same color into one "set") process to 'flip' colors when two wires of the same phase are brought into proximity. This process has been used in coloring algorithms [29] that are currently used industrially. These algorithms work well, but the runtimes can be prohibitive when run on completed Polysilicon layers (for traditional Bright Field AltPSM), and in hierarchical designs there can be cross-hierarchy interactions that preclude a successful coloring solution. In many cases, such as ASIC chips or the global routing of Microprocessors, much of the wiring of large chips is done flat (without any hierarchy), resulting in a very large wiring problem. Applying techniques like these to wiring problems that already can take several days to complete, will likely stretch the runtimes unacceptably.

MAXCUT Enabled Algorithms

It may be possible to dynamically track the coloring during the routing process, by the creation of a "dynamic MAXCUT" algorithm. At present, no such algorithm appears to exist, although one could possibly be designed for the planar graph case. In this method, colors would be tracked as wires are inserted into the design.

Likely Future Directions

These three methods all target phase colorability of the layout. None of these methods deals with the general question of "gridded" or "restricted topology" layout

styles. Uniform wire orientations are desirable for lithographic reasons beyond AltPSM. Hence phase colorability is not the only reason to pursue gridded layouts. The latest ITRS [25] sets a direction toward more gridded layout styles and rules, meaning that the most productive direction for future research may be in adapting other parts of the semiconductor design flow to the requirements of gridding.

LIST OF REFERENCES

[1] Alizadeh, F. , "Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization," *SIAM Journal on Optimization*, vol. 5, pp. 13-51, 1995.

[2] Allan, G. A. , "Yield/Reliability Enhancement using Automated Minor Layout Modifications," *Advanced Semiconductor Manufacturing 2002 IEEE/SEMI Conference and Workshop,* pp. 252-261, 2002.

[3] Allan, G. A. , "Targeted Layout Modifications for Semiconductor Yield/Reliability Enhancement," *IEEE Transactions on Semiconductor Manufacturing*, vol. 17, pp. 573-581, 2004.

[4] Allan, G. A., and Walton, A. J., "Automated Redundant Via Placement for Increased Yield and Reliability," *Microelectronics Manufacturing Yield, Reliability, and Failure Analysis III,* pp. 114-125, 1997.

[5] Allen, R. J., Hibbeler, J. D., and Tellez, G. E., inventors. International Business Machines, assignee. Use of a Layout-Optimization Tool to Increase the Yield and Reliability of VLSI Designs. no. US 6941528, Sep 6, 2005.

[6] A.S. Asratian, T.M.J. Denley, and R. Haggkvist. *Bipartite Graphs and Their Applications*, Cambridge University Press, 1998.

[7] Berman, P., Kahng, A. B., Vidhani, D., Wang, H., and Zelikovsky, A., "Optimal Phase Conflict Removal for Layout of Dark Field Alternating Phase Shift Masks," *International Symposium on Physical Design,* pp. 121-126, 1999.

[8] Berman, P., Kahng, A. B., Vidhani, D., and Zelikovsky, A., "The T-join Problem in Sparse Graphs: Applications to Phase Assignment Problems in VLSI Mask Layout," *Workshop on Algorithms and Data Structures (WADS),* pp. 25-36, 1999.

[9] Chiluvuri, V. K. R. and Koren, I., "Layout-Synthesis Techniques for Yield Enhancement," *IEEE Transactions on Semiconductor Manufacturing*, vol. 8, no. 2, pp. 178-187, 1995.

[10] Cho, J.-D., Raje, S., and Sarrafzadeh, M., "Fast Approximation Algorithms on Maxcut, k-Coloring, and K-Color Ordering in VLSI Applications," *IEEE Transactions on Computers*, vol. 47, pp. 1253-1266, 1998.

[11] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*, Wiley, 1997.

[12] de Berg, M. , Kreveld, M. v., Overmars, M., and Schwarzkopf, O., *Computationan Geometry: Algorithms and Applications,* Springer-Verlag, 1997.

[13] Dong, S., "Graph-based algorithms for floorplanning, routing, and compaction," Unversity of Illinois at Urbana-Champaign, 1994.

[14] Edmonds, J. , "Paths, Trees and Flowers," *Canadian Journal of Math*, vol. 17, pp. 449-467, 1965.

[15] Gabow, H. N. and Tarjan, R. E., "Faster Scaling Algorithms for General Graph-Matching Problems," *Journal of the ACM*, vol. 38, pp. 815-853, 1991.

[16] Garey M.R. and Johnson D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, WH Freeman and Company, 1979.

[17] Gerez, S. H. and Herrmann, O. E., "Switchbox Routing by Stepwise Reshaping," *IEEE Transactions on Computer-Aided Design*, vol. 8, pp. 1350-1361, 1989.

[18] Goemans, M. P. and Williamson, D. X., "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semi-Definite Programming," *Journal of the ACM*, vol. 42, pp. 1115-1145, 1995.

[19] Goemans, M. P. and Williamson, D. X., "Primal-Dual Approximation Algorithms for Feedback Problems in Planar Graphs," *Combinatorica*, vol. 18, pp. 37-59, 1998.

[20] Hadlock, F. O., "Finding a Maximum Cut of a Planar Graph in Polynomial Time," *SIAM Journal on Computers*, vol. 4, pp. 221-225, 1975.

[21] Harrison, N., "A Simple Via Duplication Tool for Yield Enhancement," *Proceedings of the 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems ,* pp. 39-47, 2001.

[22] Heng, F.-L. , Chen, Z., and Tellez, G. E., "A VLSI Artwork Legalization Technique Based on a New Criterion of Minimum Layout Perturbation," *Proceedings of the 1997 International Symposium on Physical Design,* pp. 116-121, 1997.

[23] Heng, F.-L. , Liebmann, L., and Lund, J., "Application of Automated Design Migration to Alternating Phase Shift Mask Design," *Proceedings of the 2001 International Symposium on Physical Design,* pp. 221-225, 2001.

[24] Huijbregts, Ed. P., van Eijndhoven, J. T. J., and Jess, J. A. G., "On Design Rule Correct Maze Routing," *European Design and Test Conference,* pp. 407-411, 1994.

[25] ITRS, International Technology Roadmap for Semiconductors, 2005.

[26] Kahng, A. B. and Pati, Y. C., "Subwavelength Lithography and its Potential Impact on Design and EDA," *Proceedings of the 36th ACM/IEEE Conference on Design Automation,* pp. 799-804, 1999.

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

[28] Kahng, A. B., Wang, H., and Zelikovsky, A., "Automated Layout and Phase Assignment Techniques for Dark Field Alternating PSM," *Proceedings of the 18th BACUS Symposium on Photomask Technology and Management,* pp. 222-231, 1998.

[29] Kim, Y. O., Lavin, M. A., Liebmann, L. W., and Weinert, G. S., inventors. International Business Machines, assignee. United States. no. 6066180, 2000.

[30] Lammers, D., “Resists not forthcoming for troubled 157 litho,” *EE Times*, vol. Jan 29, 2004.

[31] Lammers, D., “Moore's Law lives at 45 nm,” *EE Times*, vol. Jan 30, 2006.

[32] LaPedus, M., “ASML, Nikon tie in litho share race,” *EE Times*, vol. Dec 12, 2005.

[33] Lavin, M., and Liebmann, L., "CAD Computation for Manufacturability: Can We Save VLSI Technology from Itself?," *Proceedings of the 2002 IEEE/ACM International Conference on Computer Aided Design,* pp. 424-431, 2002.

[34] Lee, K.-Y., and Wang, T.-C., "Post-Routing Redundant Via Insertion for Yield/Reliability Improvement," *Proceedings of the 2006 Conference on Asia South Pacific Design Automation,* pp. 202-208, 2006.

[35] Leung, H. K.-S., "Advanced Routing in Changing Technology Landscape," *Proceedings of the 2003 International Symposium on Physical Design,* pp. 118-121, 2003.

[36] Levenson, M. D., Viswanathan, N. S., and Simpson, R. A., Improving Resolution in Photolithography with a Phase-Shifting Mask *IEEE Transactions on Electron Devices*, vol. ED-29, no. 12, pp. 1828-1836, 1982.

[37] Levinson, H. J., *Principles of Lithography*, SPIE Press, 2001.

[38] Levinson, H. J., *Principles of Lithography,* SPIE Press, 2005.

[39] Liebmann, L., Barish, A., Baum, Z., Bonges, H., Bukofsky, S., Fonseca, C., Halle, S., Northrop, G., Runyon, S., and Sigal, L., "High-Performance Circuit Design for the RET-enabled 65nm Technology Node," *Design and Process Integration for Microelectronic Manufacturing II,* 2004.

[40] Liebmann, L. W., "Layout Impact of Resolution Enhancement Techniques: Impediment or Opportunity?," *International Symposium on Physical Design,* pp. 110-117, 2003.

[41] Liebmann, L. W., Northrop, G. A., Culp, J., Sigal, L. , Barish, A., and Fonseca, C. A., "Layout Optimization at the Pinnacle of Optical Lithography," *Design and Process Integration for Microelectronic Manufacturing,* pp. 1-14, 2003.

[42] Lin, B. J., "Immersion lithography and its impact on semiconductor manufacturing," *Journal of Microlithography, Microfabrication, and Microsystems*, vol. 3, pp. 377-395, 2004.

[43] Lin, K.-K., Kale, S., and Nigam, A., "Methodology for Automated Layout Migration for 90 nm Itanium 2 Processor Design," *5th International Symposium on Quality Electronic Design,* pp. 31-35, 2004.

[44] McGrath, D., "ASML immersion tool claims industry-best NA," *EE Times*, vol. Jul 11, 2006.

[45] Moniwa, A., Terasawa, T., Hasegawa, N., and Okazaki, S., "Algorithm for Phase-Shift Mask Design with Priority on Shifter Placement," *Japanese Journal of Applied Physics, Part 1: Regular Papers & Short Notes & Review Papers*, vol. 33, no. 12B, pp. 5874-5879, 1993.

[46] Moniwa, A., Terasawa, T., Nakajo, K., Sakemi, J., and Okazaki, S., "Heuristic Method for Phase-Conflict Minimization in Automatic Phase-Shift Mask Design," *Japanese Journal of Applied Physics, Part 1: Regular Papers & Short Notes & Review Papers*, vol. 34, no. 12B, pp. 6584-6589, 1995.

[47] Ooi, K., Hara, S., and Koyama, K., "Computer Aided Design Software for Designing Phase-Shifting Masks," *Japanese Journal of Applied Physics, Part 1: Regular Papers & Short Notes & Review Papers*, vol. 32, no. 12B, pp. 5892-5899, 1993.

[48] Ooi, K., Koyama, K., and Kiryu, M., "Method of Designing Phase-Shifting Masks Utilizing a Compactor," *Japanese Journal of Applied Physics, Part 1: Regular Papers & Short Notes & Review Papers*, vol. 33, no. 12B, pp. 6774-6778, 1994.

[49] Orlova, G. L. and Dorfman, Y. G., “Finding the Maximum Cut in a Graph,” *Engineering Cybernetics*, vol. 10, no. 3, pp. 502-506, 1972.

[50] Papdimitriou, C. H., and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity,* Dover, 1982.

[51] M. Reinhart. *Automatic Layout Modification*, Kluwer Academic Publishers, 2002.

[52] Sahni, S., and Gonzalez, T., “P-Complete Approximation Algorithms,” *Journal of the ACM*, vol. 23, no. 3, pp. 555-565, 1976.

[53] Scheffer, L. K., "Physical CAD Changes to Incorporate Design for Lithography and Manufacturability," *Proceedings of the 2004 conference on Asia South Pacific Design Automation,* pp. 768-773, 2004.

[54] N. Sherwani . *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, 1999.

[55] Shi, X., Hsu, S., Chen, J. F., Hsu, M., Socha, R. J., and Dusa, M. V., "Understanding the forbidden pitch and assist feature placement," *Proceedings of the 21st BACUS Symposium on Photomask Technology,* pp. 968-979, 2002.

[56] Skiena, S. S. *The Algorithm Design Manual*, Springer-Verlag, 1998.

[57] Torres, J. A., and Chow, D., "RET Compliant Cell Generation for sub-130nm Processes," *Design, Process Integration, and Characterization for Microelectronics,* pp. 529-539.

[58] A.K.-K. Wong. *Resolution Enhancement Techniques in Optical Lithography*, SPIE Press, 2001.

[59] Wong, A. K.-K., “Microlithography: Trends, Challenges, Solutions and Their Impact on Design,” *IEEE Micro*, vol. 23, pp. 12-21, 2003.

[60] Xu, G., Huang, L.-D., Pan, D. Z., and Wong, M. D. F., "Redundant-Via Enhanced Maze Routing for Yield Improvement," *Proceedings of the 2005 conference on Asia South Pacific Design Automation,* pp. 1148-1151, 2005.

[61] Yamada, Y., “Exposure Tool Strategy for 90nm - 65nm Production,” *Semiconductor Fabtech*, vol. 18, 2003.

[62] Yan, C., “ProMOS to spend $5 billion on two 300-mm fabs,” *EE Times*, vol. Sep 1, 2006.

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

BIOGRAPHICAL SKETCH

Kevin McCullen received a Bachelor of Science in electrical engineering from Purdue University in West Lafayette, IN, in 1982. In 1991 he received a Master of Science degree in electrical and computer engineering from Clarkson University in Potsdam, NY. His master's thesis topic was queuing network models for RISC and CISC performance analysis. Since 1982, he has been employed by International Business Machines, working in Computer Aided Design software. He has worked in the areas of chip floorplanning, placement, routing, manual layout, design for manufacturability and technology migration of VLSI designs. He has filed 12 US Patents, and published four papers. He will receive a PhD in computer engineering from The University of Florida in 2006.