UNIVERSITY OF MINNESOTA

This is to verify that I have examined this copy of a doctoral thesis
by

Tianpei Zhang

and have found that it is complete and satisfactory in all aspects,
and that any and all revisions required by final
examining committee have been made.

Professor Sachin S. Sapatnekar

Name of the Faculty Advisor

Signature of the Faculty Advisor

Date

GRADUATE SCHOOL

Routing Issues in Nanometer-scale Integrated Circuits

A PhD Dissertation
Submitted to the Faculty of the Graduate School
of The University of Minnesota
By

Tianpei Zhang

in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy in Electrical and Computer Engineering

Professor Sachin S. Sapatnekar, Advisor

September 2006

ABSTRACT

As integrated circuits (ICs) have scaled into nanometer dimensions, operating at gigahertz frequencies, interconnects have become critical in determining system performance, reliability, and manufacturability. This thesis presents new algorithms dealing with several important aspects of VLSI interconnects optimization and prediction, namely, noise reduction in global routing, temperature-aware global routing for three-dimensional (3D) ICs, buffer insertions issues for global interconnects in structured ASICs, and chemical mechanical polishing (CMP)-aware routing for optimized dummy metal fills.

The issue of crosstalk noise during global routing is first targeted in this work, employing both buffers and shielding lines to mitigate this noise. A dynamic programming-like algorithm is utilized to efficiently reduce crosstalk noise under the constraints of routing and buffering resources. The algorithm is presented under a flexible supply network architecture so that existing power wires are utilized as shields. The noise level is evaluated with the simple yet effective metric proposed by Devgan. It is proposed that Devgan's metric, which is pessimistic and is known to have limited accuracy, shows excellent fidelity properties, and a noise margin inflation technique is developed to compensate for its pessimism.

The second issue studied in this thesis is related to electrothermally-conscious global routing for 3D ICs. 3D ICs are built by stacking multiple layers of active devices, which provides the potential for reduced interconnect delays. However, this technology is plagued with thermal problems. Our global routing algorithm, in addition to solving the routing problem in all three dimensions, plans thermal vias and thermal wires to establish good heat conduction paths. The algorithm utilizes sensitivity analysis and linear programming techniques to judiciously allocate the usage of thermal vias and thermal wires, and iteratively resolve contentions between signal routing, thermal vias and thermal wires, while simultaneously managing thermal tradeoffs.

The next part of the thesis considers buffering global interconnects in structured ASICs. This is a relatively new technology that creates extremely regular layouts, primarily in order to avoid a

number of manufacturability problems that arise in the nanometer regime. As buffers need to be predistributed and prefabricated in structured ASICs, we develop an accurate buffer distribution prediction algorithm based on Rent's rule, statistical routing estimation and a simplified buffer insertion model. Based on the prediction of buffer distributions, we can prefabricate a series of structured ASIC templates, so that preplaced buffers in each template can accurately and economically satisfy the buffering needs of a whole family of implemented circuits fitting into the template.

A major consideration in the design of nanoscale integrated circuits is related to the insertion of dummy fills in each metal layer, in order to guarantee planarity during CMP. The final part of the thesis develops a manufacturability-aware routing flow that optimizes the amount of CMP dummy fills. Dummy fills can affect circuit performance and also result in mask data explosion. Our routing algorithm presents an effort in design phase to address the manufacturability issue. We propose a novel dummy fill metric, and incorporate the CMP dummy filling model throughout the routing process. By employing CMP-aware heuristics in global routing and layer assignment stages, our routing algorithm can optimize the amount of dummy fills needed in post-layout processing.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Sachin Sapatnekar, for giving me the invaluable opportunity to work on challenging and extremely interesting projects in the field of Design Automation. I am truly indebted to his persistent help, encouragement and guidance throughout my Ph.D. research work. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank to Professor Kia Bazargan, Professor Chris Kim and Professor Pen-Chung Yew for their helpful and resourceful discussions on research project, as well as agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript. I owe many thanks to Guoqiang Chen, the mentor of my internship at Rio Design Automation. I have learnt a lot from him on how to bridge the theoretic and practical sides in EDA industrial development.

My colleagues at VLSI Electronic Design Automation (VEDA) Lab have enriched my graduate life in many ways and deserve a special mention: Hongliang Chang, Yang Feng, Brent Goplen, Haitian Hu, Jiang Hu, Shirirang Karandikar, Mahesh Ketkar, Sanjay Kumar, Vidyasagar Nookala, Anita Pratti, Haifeng Qian, Venkat Rajappan, Rupesh Shelar, Jaskirat Singh, Haihua Su, Anup Sultania, and Yong Zhan.

I owe my deepest thanks to my family - my mother and father who have always stood by me and guided me through my pursuit of Ph.D. degree. Words cannot express the gratitude I owe them. I am grateful to my wife, Ximing Yang for her support and understanding during my studies, and most of all, for making my life more meaningful.

I would like to acknowledge financial support from the National Science Foundation and

Semiconductor Research Corporation for funding parts of my research and providing financial support to attend conferences.

It is impossible to remember all, and I apologize to those I've inadvertently left out. Lastly, thank you all!

DEDICATION

To my parents and my wife.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Interconnect Challenges

As Very Large Scale Integrated (VLSI) fabrication technology continues to scale down to below sub-100nm dimensions, integrated circuits (ICs) have undergone dramatic progress, as characterized by Moore's law [84], which predicts that the transistor density of semiconductor chips will double every eighteen months in the foreseeable future. It is projected by 2005 International Technology Roadmap for Semiconductors (ITRS'05) [8] that this trend will continue for the next decade, summarized in Table 1.1, and there will be over three billion transistors integrated on a single chip operating at a clock frequency of 39GHz at the 22nm technology node by year 2016.

Tech. Node (nm)	Year	Num. Transistors	Num. Wiring Level	f (MHz)	V_{dd} (V)	size mm^2
90	2005	193M	11	5204	0.9	111
68	2007	386M	11	9285	0.8	140
59	2008	386M	12	10972	0.8	111
45	2010	773M	12	15079	0.7	140
36	2012	773M	12	20065	0.7	88
32	2013	1546M	13	22980	0.6	140
22	2016	3092M	13	39683	0.5	140

Table 1.1: Trends in IC technology parameters [8].

With technology scaling, interconnect delays become a larger fraction of the clock frequency,

as wires become thinner (and hence,more resistive) and the proportion of global wires increases. Both trends cause wires to be a major limiting factor and bottleneck in determining system performance. Therefore interconnect optimization is a crucial task in meeting the high-speed transmission needs of chips under the scaling of feature sizes. Apart from performance issues, it is growing more difficult to resolve all interconnect-related manufacturability issues in nanoscale technologies, and another great challenge facing interconnect design is how to plan interconnects so that a design can be fabricated with reasonable cost and high yield.

Moreover, new design paradigms have begun to emerge, including three-dimensional integrated circuits (3D ICs) and structured ASICs, each of which brings forth new challenges in interconnect design. 3D ICs are fabricated by stacking multiple layers of conventional (single wafer) ICs into a monolithic structure and electrically interconnecting these layers with inter-layer vias. The more advanced versions of this technology thin the wafers prior to stacking, so as to reduce wirelengths, and consequently, delays, in the third dimension. This new circuit style can greatly enhance system performance and pack more transistors per volume. Nevertheless, routing for 3D ICs is inherently more complex due to the challenges of handling more dimensions than their two-dimensional (2D) counterpart. Moreover, thermal effects in 3D ICs are more pronounced than 2D ICs due to higher power density and poor thermal conduction, and this problem must be addressed in 3D routing tools to generate high performance and reliable designs. Structured ASICs are another new technology, where a chip is composed of regular arrays of prefabricated standard building blocks. With fixed mask structures, they show good manufacturability and lower non-recurring engineering (NRE) costs, but the the highly regular nature implies that new issues must be solved for the optimal design of interconnects in structured ASICs.

Within the interconnect design flow, one of the most important steps is routing, in which pins of signal nets are connected for a given placement, so that signals can be transmitted from a source pin to a set of sink pins. Physical on-chip resources such as routing space, routing layers and vias, are allocated during this phase and a path is found for each net. The quality of the routing solution can directly affect the speed, congestion, noise and power consumption of a chip,

and hence this step plays an important role in determining circuit performance.

The most basic form of this problem is single net routing, which builds a rectilinear tree that spans all pins in a given net. The objective during this tree construction is usually to minimize wire length and to control signal delays to meet timing requirements. Over and above this, buffer insertion may be used for interconnect optimization, to effectively reduce signal delay and block noise propagation for long wires; therefore buffers are widely used in modern interconnect design and their insertion is usually addressed during routing. The overall problem is, however, more complex: a state-of-the-art routing system must simultaneously route multiple nets, during which all nets compete for limited routing resources, while wiring congestion and other performance related issues need to be optimized. This has been proved to be a difficult problem since a simple form of it, routing a set of two-pin nets under congestion constraints, is NP-complete [71].

The objectives that must be optimized during routing are growing increasingly complex in the nanometer design regime. Apart from traditional metrics such as timing and wire length, routing is facing a new set of challenges such as routability, noise, thermal-related problems, manufacturability and the prediction of routing resource allocations.

Crosstalk noise increasingly affects the performance of VLSI circuits due to the decreasing wire separation and higher aspect ratio of metal wires, which leads to an increased contribution of the coupling capacitance to the total capacitance. Since routing is a major step in determining the physical routes of wires, and in planning noise avoidance resources, such as shields and buffers, among nets, it is important to integrate the analysis and optimization of crosstalk noise into routing in order to maintain signal integrity, and this approach is discussed in the thesis.

Another important issue related to routing is the fast prediction of routing properties, i.e., to predict routing results, such as the wire length distribution and congestion map of a design, without actually performing routing. This prediction technology can be incorporated to various design stages such as floorplanning and placement to compare superiority of a design; under some circumstances, routing prediction is even crucial in obtaining the circuit properties, which is otherwise impossible to be acquired, for some special circuit styles such as structured

ASICs.

Another significant challenge plaguing modern nanometer-scale ICs is manufacturability, which demands new methodologies to handle the corresponding design constraints. One such issue is related to metal fill insertion to maintain planarity during chemical-mechanical polishing (CMP). Routing, as a major phase to distribute metal wires in design, plays a key role in this aspect of design for manufacturability (DFM). A manufacturing-friendly routing solution can greatly relieve the post-layout processing efforts, and better keep designer's intent, so that the system performance is less affected.

Therefore, interconnect performance and manufacturability become ever increasing challenges with the scaling of fabrication technologies, and modern routing technologies must be tailored to solve the corresponding problems.

1.2 Contributions of this Thesis

This thesis addresses performance issues in routing, including signal integrity, routing resource prediction and thermal problems in interconnect optimization, as well as manufacturability-related issues such as buffer prediction and dummy fill optimization. These issues are addressed under various design paradigms, including regular ASICs, structured ASICs, and 3D ICs. The major contributions of this thesis are as follows:

- In Chapter 3, we propose a method for incorporating crosstalk reduction criteria into global routing under a broad power supply network paradigm. This method utilizes power/ground (P/G) wires as shields between signal wires to reduce capacitive coupling, and simultaneously employs buffers to block noise propagation, while considering the constraints imposed by limited routing and buffering resources. Our noise calculations are based on Devgan's noise metric [40], and our work demonstrates, for the first time, that this metric shows good fidelity on average. An effective noise margin inflation technique is also proposed to compensate for the pessimism of Devgan's noise metric. Experimental results shows our algorithm achieves better noise reduction than those methods considering only buffer insertion, or only shield

insertion after buffer planning.

- In Chapter 4, a novel temperature-aware 3D global routing algorithm is proposed. 3D ICs provide an attractive solution for improving circuit performance, but suffer from escalated thermal problems due to a significant amount of heat per unit volume. Our temperature-aware 3D global routing algorithm follows a electrothermally-conscious design methodology, and reduce chip temperature with insertion of thermal vias and thermal wires to lower the effective thermal resistance of the material. Since thermal vias and thermal wires take up lateral routing space, our algorithm utilizes adjoint sensitivity analysis and linear programming (LP) to judiciously allocate their usage, and iteratively resolve contention between routing and thermal vias/wires. Experimental results show that our temperature-aware routing algorithm can effectively reduce the peak temperature and alleviate routing congestion in 3D ICs.

- Chapter 5 introduces a new buffer insertion and predistribution methodology in structured ASIC designs, based on a novel routing prediction technique. Due to the regularity that is enforced in this style of designs, buffers must be prefabricated on structured ASICs. The precise choices of the number and distribution of the buffers are critical, and are closely related to the outcome of routing. Moreover, since the underlying fabric of structured ASICs must be, to some extent, design-independent, it is essential to determine this distribution based on very coarse design information. Our solution to this problem is to statistically predict the routing properties with Rent's rule, then estimate the buffer distribution based on the routing properties and a simplified buffer insertion model. Hence a series of structured ASIC templates can be prefabricated with buffers preplaced according to the buffer distribution estimation results. Experiments on benchmarks prove that our buffer prediction is accurate and economical, and the preplaced buffers in these structured ASIC templates can satisfy the buffering demand in practice.

- In chapter 6, we develop a CMP-aware routing algorithm that minimizes the amount of

dummy metal fills in post-layout processing. We propose a novel dummy fill metric which can estimate the amount of dummy fill without resorting to the repeated solutions of high complexity Linear Programming (LP) instances for each candidate topology. By embedding the dummy fill model and metric during the routing process, our algorithm can consciously spread wires laterally and assign wires to appropriate metal layers so as to get a more "uniform" metal wire distribution through the routing phase. Our experimental results show that our routing algorithm, based on this metric, minimizes the amount of CMP dummy fills, and is more manufacturable, as compared to a conventional router.

Part of the research has been included in the papers [1, 122, 123, 124, 125, 126, 127].

Chapter 2

Preliminaries

This chapter presents some background material on routing that is used through the rest of this thesis. Section 2.1 begins by presenting some basic definitions and an overview of a routing flow, and is followed by Section 2.2, which discusses the concept of a routing graph. Next, buffer insertion models are overviewed in Section 2.3, and performance metrics for delay, crosstalk and temperature are presented in Section 2.4. Finally, the concept of Rent's rule, which is used to predict interconnect distributions, is detailed in Section 2.5.

2.1 Basic Definitions and the Overall Routing Flow

A *Net* N consists of a set of k electrically equivalent pins $\{s, p_1, p_2, ..., p_{k-1}\}$, that must be connected by wires, of which s is the source and $p_1, p_2, ..., p_{k-1}$ are the sinks. The practical routing problem in a VLSI circuit deals with a set of nets, $\mathcal{N} = \{N_1, N_2, \ldots\}$, the detailed paths of whom will be determined so that *routability* is achieved and other performance metrics are optimized: this is referred to as a *multi-net routing* problem [60]. A routing solution is said to be *routable* if at any location on the chip, the amount of routing resources utilized is no more than the available resources.

The typical strategy for solving a complex routing problem is to tackle it through several stages. Traditionally, this has been done by dividing the problem into the *global routing* and *detailed*

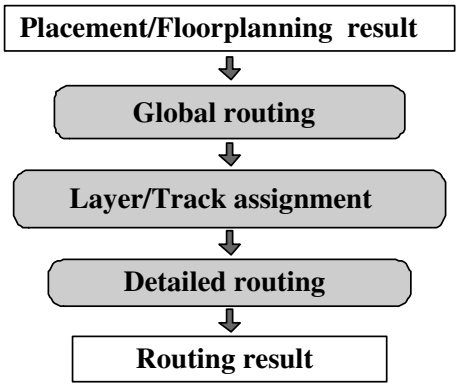


Figure 2.1: Overall flow of a typical modern routing system.

routing stages [100], with a layer/track assignment stage in between these two stages. In the global routing stage, the entire routing area is divided into an array of rectangular subregions, each of which may contain tens of routing tracks in each dimension. The pins of the nets in the design are mapped to subregions, and for each net, a route is constructed as a set of subregions within which a physical connection can be embedded. Usually, Steiner tree [30] and maze search [100] techniques are used to generate the global route for a net. In the detailed routing stage, a detailed physical connection is constructed for each net within the global routing subregions determined in previous step. To better utilize global routing information and efficiently address problems arising from signal delay, crosstalk noise and process constraints, a track/layer assignment stage is applied between global and detailed routing [12, 19, 68]. In this intermediate stage, the metal layer and routing tracks for net segments are determined to optimize performance-related metrics. Figure 2.1 depicts the overall flow for a typical modern routing system. The input to the router is the result of placement/floorplanning, where the pin locations have been determined. This feeds through the three routing stages described above, to generate a routing result. The role of global routing is to plan the approximate routing paths for subsequent stages, and hence it has a significant impact on system performance metrics such as wirelength, routability, and signal integrity, and therefore, this thesis places a major focus on the global routing stage.

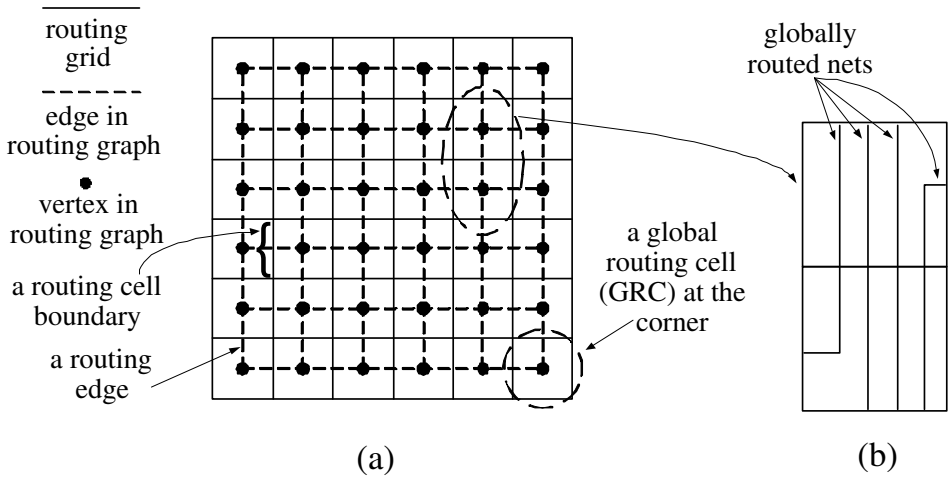


Figure 2.2: (a) A routing grid and its corresponding global routing graph. (b) An inset view of four globally routed nets simultaneously passing through the boundary between two GRCs.

2.2 Global Routing Graph

2.2.1 Global Routing Graph for 2D ICs

A fundamental notion in global routing is the *global routing graph*, which plays a central role in global routing resource planning and is a critical ingredient of the problem formulation. For a regular 2D IC, as shown in Figure 3.1(a), our global routing model tessellates the entire chip into an array of grid cells, referred to as the *routing grid*; each grid cell is then referred to as a *global routing cell (GRC)*. The dual graph of the routing grid tessellation is the *global routing graph* $G = (V, E)$. The elements of V and E are shown in Figure 2.2(a) as darkened dots and dashed lines, respectively. Each vertex $v \in V$ represents a GRC and each edge $e \in E$ corresponds to a boundary b_{ij}, which connects two neighboring GRCs i and j. The *grid length* L_e for an edge e is defined as the center-to-center distance between two neighboring GRCs.

During the global routing stage, the pins of a net $N_i \in \mathcal{N}$ are first mapped to the corresponding GRCs, thus we can obtain a set of vertices $VI_i = \{v_{i,0}, v_{i,1}, \ldots\} \subseteq V$, which are associated with those GRCs. The routing problem for a net $N_i \in \mathcal{N}$ is to find an additional set of vertices $VA_i \subseteq V$, which are called Steiner vertices, and a set of edges $E_i = \{e_{i,0}, e_{i,1}, \ldots\} \subseteq E$, which are referred to as *routing path*, to form a rectilinear minimum spanning tree $T_i = (V_i, E_i)$, where

$V_i = VI_i \cup VA_i$, so that some performance-related metric is optimized.

As multiple nets are routed over the routing graph, a common boundary between two neighboring GRCs may be crossed by wires belonging to different nets. In terms of the graph representation, this implies that multiple routing trees can utilize a common edge $e \in E$. Figure 2.2(b) depicts such a scenario: four globally routed nets simultaneously pass through the boundary (corresponding to an edge in the routing graph) between two neighboring GRCs. Due to routing space limitations on the boundary corresponding an edge e, we require that $W_e \leq C_e$, in which W_e is the number of nets passing the edge e, defined as *edge usage*, and C_e is the upper bound of available routing tracks over edge e, referred to as *edge capacity*. A violation of this requirement results in *overflow*, which is numerically defined as $W_e - C_e$. The most fundamental requirement for a global routing solution is that all of the nets should be routed so that there is no overflow along any routing graph edge.

2.2.2 Global Routing Graph for 3D ICs

In three-dimensional (3D) ICs, devices are fabricated on a number of stacked active layers. Within each device layer, devices can be interconnected with traditional interconnect wires and vias. Connections between active layers are facilitated by vertical interconnect vias that span through two or more layers, providing a means for electrically connecting wires in those layers; we refer to such vias as *interlayer vias*.

To better characterize the unique interconnect resources in 3D ICs, we introduce a 3D global routing graph which is illustrated in Figure 2.3. We first build a regular 2D global routing graph for each active layer as in Section 2.2.1. Next, by stacking the global routing graphs of all layers and connecting each pair of vertically neighboring routing nodes with a vertical routing edge, we build a *3D global routing graph*, which is shown by the darkened dots and dashed lines in Figure 2.3. There are two classes of edges in the graph:

- Each lateral edge in the 3D global routing graph is derived from its counterpart in regular 2D global routing graph, and follows the same routing resource constraints as in Section 2.2.1;

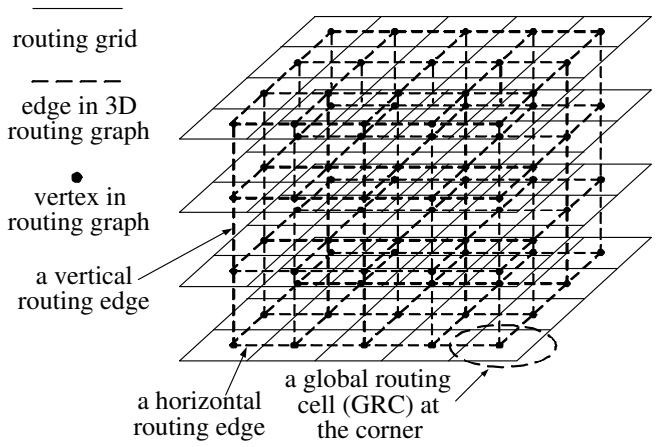


Figure 2.3: A 3D global routing graph for a 3D IC with four active layers.

any overflow along a lateral edge is referred to as *lateral overflow*.

- Each vertical edge in the 3D routing graph corresponds to the interlayer vias connecting two vertically neighboring GRCs. As interlayer vias can only use the white space that is not occupied by devices while passing active layers, there is an upper limit on the number of interlayer vias for each vertical edge in the routing graph, referred to as *interlayer via capacity* U_i. We require $V_i \leq U_i$, in which V_i is the actual number of interlayer vias through GRC i, otherwise, we obtain *vertical overflow*.

The problem of global routing in 3D ICs involves finding the spanning trees of all nets on the 3D global routing graph, so that there is no lateral and vertical overflow, and some specified performance-related metrics are optimized.

2.3 Buffering Model for ASIC and Custom Design

Buffer insertion has become a crucial step in modern interconnect design to meet the stringent timing and noise requirements [4, 30, 98]. The insertion of these buffers divides long wires into shorter segments, and to the first order, it reduces delay to be approximately linear in the wire length, as opposed to the quadratic length dependence of an unbuffered line. Another benefit of buffer insertion is its ability to recover noise margin and block noise propagation [5]. These

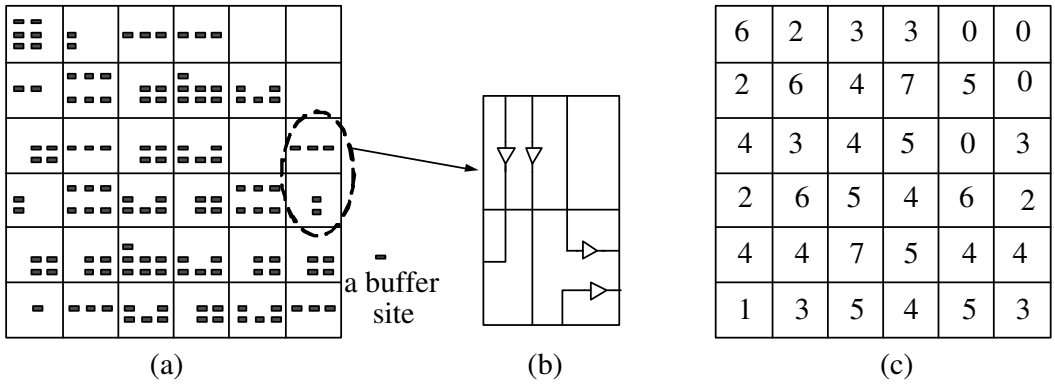


Figure 2.4: (a) A design is tessellated into tiles, and a total of 131 buffer sites are interspersed within the tiles. (b) An inset view that four out of five buffer sites in two neighboring tiles are actually implemented to be buffers and inserted to nets. (c) Buffer capacity is listed for each tile in the corresponding tessellation of (a).

attributes make buffer insertion a pervasive practice in modern design.

In ASIC and custom design, buffer insertion is generally performed at the back end of the design process. However, to manage large number of buffers and achieve high performance on critical global nets, the distribution of buffers through the layout must be planned early in the design flow. Various models for buffer placement in ASIC and custom design have been proposed in literature, prominent among which are buffer block planning [31], in which buffer blocks are placed *between* the building blocks of the circuit, and distributed buffer model [5], in which buffers are interspersed *within* each building block, with their exact location and insertion being undetermined until later in the design process. The latter approach has several advantages over the former in terms of reduced congestion and increased flexibility [5]. Therefore this distributed buffer model is adopted in our work, and described here in more detail.

In the distributed buffer model, designers can plan and dispersed buffer resources across a design by inserting *buffer sites*, i.e., physical areas that can be utilized to implement buffers, into building blocks. When a buffer site is actually assigned to a net, a buffer gate from library is then physically selected and inserted. For planning purposes, we can tile the chip region. Figure 2.4(a) shows a design is tessellated into a two-dimensional 6×6 array of tiles, and a total of 131 buffer sites are interspersed within the tiles. Further, Figure 2.4(b) illustrates a scenario that four out of five buffer sites in two neighboring tiles are actually assigned to nets and buffers are inserted

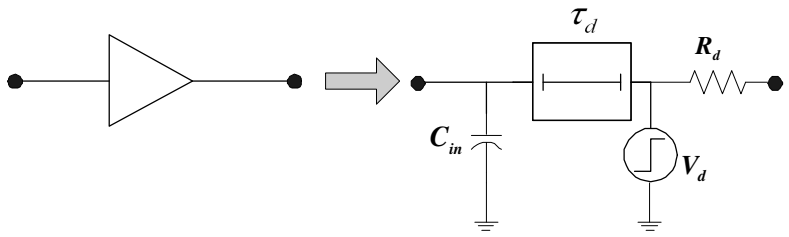


Figure 2.5: Linear model for a buffer.

to global interconnects. As buffer planning is closely related to global routing, in practice, this tiling scheme often employs the global routing grid which is defined in Section 2.2.1. For a tile i, the number of buffer sites allocated is referred to as *buffer capacity*, and denoted as B_i, which corresponds to the maximum number of buffers that can be utilized within this tile. Figure 2.4(c) lists the buffer capacity of each tile for the tessellated design in Figure 2.4(a). If the number of utilized buffers is b_i, then $b_i \leq B_i$ must be satisfied, otherwise we have *buffer overflow*.

Electrically, a buffer is modeled by an equivalent linear circuit shown in Figure 2.5. This circuit includes a buffer input capacitance C_{in} that loads the previous interconnect stage, an intrinsic delay τ_d associated with the buffer, and an interface with the next interconnect stage, consisting of an ideal voltage source V_d and an effective driver resistance R_d. Based on this model, various buffer insertion algorithms [3, 79, 102, 110] have been proposed to optimize net delay. However, it is observed in [31] that the buffer position along an interconnect can be rather flexible without inducing significant timing degradations. This is supported in [42, 99], where the authors observed that there is a maximum distance between two consecutive buffers so that the timing performance and sharp slew rate can be ensured. Therefore, like [5, 42, 99], we employ a simple yet efficient buffer insertion requirement: a maximum total interconnect length L driven by a buffer (gate) is enforced to control the interconnect delay and slew rate, and this maximum length is referred to as the *critical length*. In practice, [42] shows that, for a high-end microprocessor design in 0.25μm technology, consecutive buffers are separated by at most 4500μm. In terms of interconnect structure, our simple buffer insertion requirement is illustrated in Figure 2.6 for the single fanout and two-fanout case as examples. In Figure 2.6(a), we simply require the separation between two buffers is no more than L as the driving buffer has only one fan-out, while Figure 2.6(b)

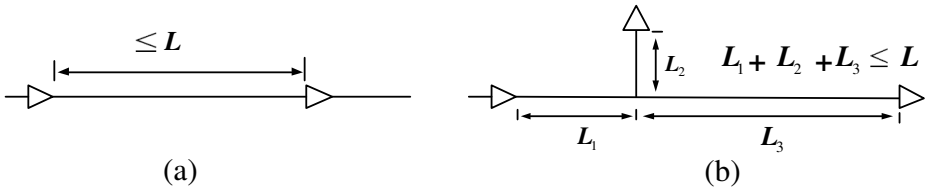


Figure 2.6: Buffer insertion requirement in terms of buffer separation: (a) Single fanout case. (b) Two-fanout case.

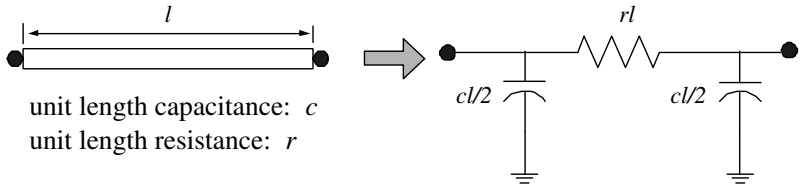


Figure 2.7: π RC model of a wire segment.

depicts the case that the driving buffer has two fan-outs, and we require the total interconnect length between the driving buffer and its downstream buffers (which is $L_1 + L_2 + L_3$) should be less than or equal to L.

2.4 Interconnect Performance Metrics

2.4.1 Interconnect Delay Modeling

To calculate the signal delay along an interconnect, we adopt the π RC model for each wire segment, as depicted in Figure 2.7. Based on this, the equivalent RC circuit of a interconnect tree is then extracted, and its delay is calculated. For a tree-structured RC circuit, the Elmore delay model [43] has been widely used due to its simple form. For a tree-structured net N, denote the driver resistance at source s_0 as R_d, the delay from s_0 to a sink node s_i, $i = 1, 2 \ldots$, can be expressed as:

$$T_{s_0 \to s_i} = R_d \cdot C_0 + \sum_{edge\ e_{jk} \in P_i} \left(\frac{r c l_{e_{jk}}^2}{2} + r l_{e_{jk}} \cdot C_k \right) \tag{2.1}$$

where r and c are, respectively, the wire resistance and capacitance per unit length; P_i is the path from source s_0 to sink s_i; $l_{e_{jk}}$ is the length of edge e_{jk}, and we denote k as the downstream

node; C_k is the total downstream capacitance (including sink input capacitance and interconnect capacitance) for a node k in the tree. The Elmore delay is used widely in delay estimation due to its simple form, which is especially amenable for optimization. However, this metric fails to capture some important factors such as resistive shielding [95], hence it tends to overestimate the delay, and can be shown to give an upper bound for delay [50]. However, a good property about Elmore delay is its high fidelity [18], i.e., an optimal or near-optimal solution under Elmore delay will also be near-optimal according to more accurate metric. This property justifies it usage as an effective delay estimation metric for identifying good interconnect structures.

2.4.2 Devgan's Noise Metric

Various transient analysis techniques can be used to estimate crosstalk noise in an interconnect network. Devgan proposed a fast closed-form metric in [40] for coupled noise estimation between wires, in which various circuit electrical properties, such as distributed coupling capacitance, aggressor slew rate and wire resistance, are taken into consideration. It estimates the noise using a formula that resembles the Elmore delay model discussed in last section, which makes it amenable to being embedded in the optimization phase of physical design [4, 22, 77, 82].

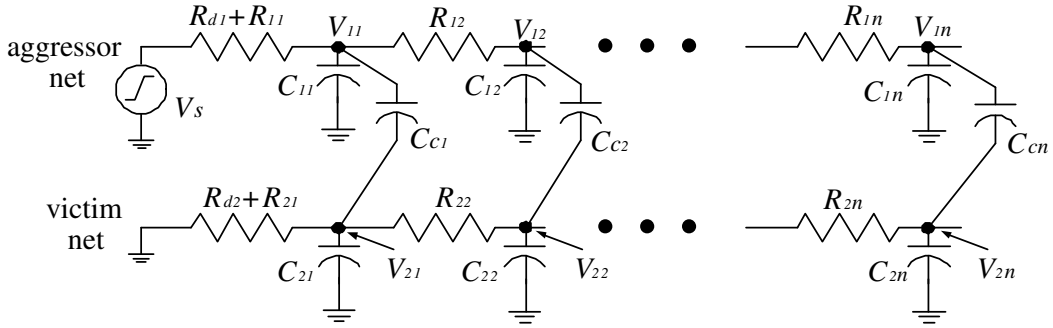


Figure 2.8: Circuit schematic of capacitive coupled interconnect network consisting of two nets.

Devgan's interconnect noise estimator first extracts the interconnect resistance and the ground and coupling capacitance for a set of coupling nets. This may be performed with a π model for the interconnect, which results in a coupled RC ladder network. An example consisting two coupling nets is shown in Figure 2.8. Both nets are divided into n segments, and for each segment,

we can extract R_{ij}, C_{ij}, and C_{cj}, $i = 1, 2$ and $j = 1, \ldots, n$, which are the segment resistance, segment ground capacitance, and coupling capacitance between two segments, respectively. The upper RC ladder represents the *aggressor net* (indexed as 1) which is driven by a ramp voltage V_s and has a driver resistance R_{d1}, whereas a noise pulse is induced in the lower one, which is otherwise quiet, due to the aggressor net. The lower net is the *victim net* (indexed as 2) whose driver resistance is R_{d2}. Building the nodal analysis (NA) equation [93], and after some simplification, we can relate the node voltage vector $\mathbf{V}_2 \in \mathcal{R}^{n\times 1}$ for the victim net to the aggressor input voltage V_s as:

$$[(s\mathbf{C}_2 - \mathbf{A}_{22}) - s\mathbf{C}_c(s\mathbf{C}_1 - \mathbf{A}_{11})^{-1}s\mathbf{C}_c]\mathbf{V}_2 = -s\mathbf{C}_c(s\mathbf{C}_1 - \mathbf{A}_{11})^{-1}\mathbf{B}_1 V_s \tag{2.2}$$

in which $\mathbf{A}_{11}$ and $\mathbf{A}_{22}$ are the node resistance matrices of the aggressor and victim net, respectively; $\mathbf{C}_i = diag(C_{ij} + C_{cj})$ for $i = 1, 2$ and $j = 1, 2, \ldots, n$ and $\mathbf{C}_c = diag(-C_{ci})$ for $i = 1, 2, \ldots, n$; $\mathbf{B}_1$ is the resistance vector from the input to all voltage nodes on the aggressor net. Given an infinite ramp voltage at the aggressor input, the node voltages at the victim net monotonically increase towards their final value. Utilizing the final value theorem, we can obtain the maximum steady-state node voltages at the victim net as:

$$\mathbf{V}_{2,max} = -\mathbf{A}_{22}^{-1}\mathbf{C}_c\mathbf{A}_{11}^{-1}\mathbf{B}_1\dot{u} \tag{2.3}$$

where $\dot{u}$ is the aggressor input slope. Furthermore, if we interpret $\mathbf{C}_c\mathbf{A}_{11}^{-1}\mathbf{B}_1\dot{u}$ as a vector of noise current $\mathbf{I}_c$, equation (2.3) can be further simplified as:

$$\mathbf{V}_{2,max} = -\mathbf{A}_{22}^{-1}\mathbf{I}_c \tag{2.4}$$

For a typical tree-structured interconnect of the victim net, computation of $\mathbf{V}_{2,max}$ with equation (2.4) is very similar to that of Elmore delay, and more details can be found in [40].

The limitation of Devgan's noise metric lies in the fact that it assumes an infinite ramp

voltage at aggressor input, hence it computes only an upper bound on the circuit noise, and is often overly pessimistic, especially for long wires and fast slew rate [58, 72]. Despite its pessimism, the work in this thesis shows for the first time that, Devgan's metric has good fidelity, in the sense that the relative ordering of the noise value at two nodes,when evaluated with Devgan's metric, is very likely to be the same as that determined by accurate noise simulations. This property implies that Devgan's metric can work as a simple yet effective metric in noise value comparison and pruning of noise protection solution.

2.4.3 Thermal Analysis in ICs

As technology scales, thermal analysis becomes more important as power density is getting higher and circuit performance and reliability are more affected by thermal effects. Thermal analysis for ICs typically involves finding the temperature across the chip, given the power distribution and thermal boundary condition. The time constant for on-chip heat conduction is orders of magnitude larger than the clock cycle in a modern IC, therefore we generally study the steady-state thermal analysis, and describe it with the following Poisson equation:

$$\begin{aligned} & k(x,y,z)\nabla^2 T(x,y,z) + g(x,y,z) = 0 \\ \text{subject to boundary condition}: \quad & k(x,y,z)\frac{\partial T(x,y,z)}{\partial n_i} + h_i T(x,y,z) = 0 \end{aligned} \tag{2.5}$$

where T is the temperature, k is the thermal conductivity, g is the power density in chip, and n_i and h_i are the outward normal direction and heat transfer coefficient for surface i, respectively. The Poisson equation for thermal analysis is usually solved numerically using the finite difference method (FDM), or the finite element method (FEM) as in [78, 47, 109], or the Green function method as in [23, 121]. The FDM is fast and amenable to obtain temperature and sensitivity information, hence is employed for thermal analysis in this thesis. Typically, the substrate is tessellated to be a three-dimensional (3D) array of cells and then modeled as a lumped circuit network, hence the temperature can be obtained by solving the nodal equation of the network

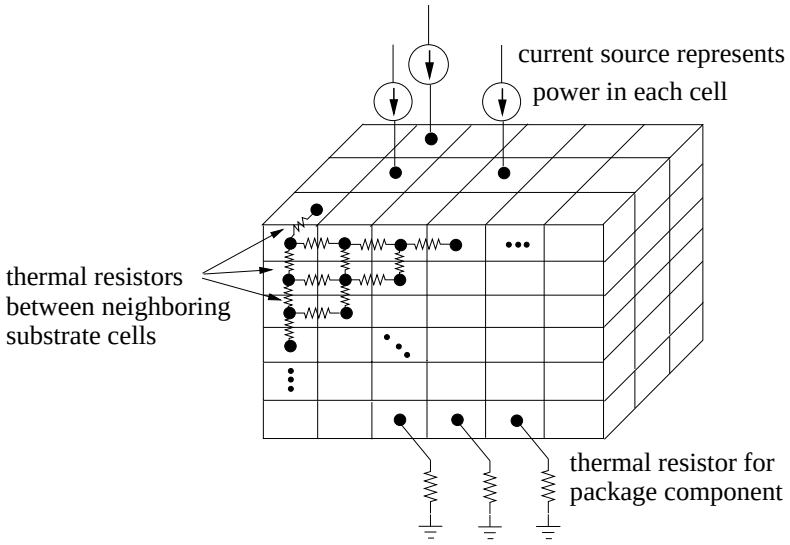


Figure 2.9: Heat transfer in substrate can be modeled as a 3D circuit network composed of thermal resistors and current sources.

numerically. Figure 2.9 illustrates the 3D tessellation of a substrate and the modeling of the relevant elements in a thermal circuit. Representing each cell in the grid as a node, we can discretize equation (2.5) and build a nodal analysis (NA) equation [93] for a thermal circuit as:

$$\mathbf{MT} = \mathbf{P} \tag{2.6}$$

in which $\mathbf{M}$, $\mathbf{T}$ and $\mathbf{P}$ are the NA coefficient matrix, the temperature profile vector and power dissipation vector for each grid cell. The size of matrix $\mathbf{M}$ can be large, but due to the fact it is sparse, and symmetric and positive definite (SPD), this linear equation can be solved efficiently with an iterative linear solver such as LASPack [104].

2.4.4 Adjoint Sensitivity Analysis

Incremental sensitivity is defined as the partial derivative of a circuit response with respect to a circuit parameter of interest, which is valid in a small range around the nominal value of the parameter. Adjoint sensitivity analysis can calculate the incremental sensitivity of a single circuit response with respect to multiple parameters efficiently [93]. This advantage makes adjoint sensitivity analysis a natural choice in evaluating the sensitivity of the temperature change at a hot

spot with respect to thermal resistance changes at various locations in 3D thermal circuit analysis process.

Details about adjoint sensitivity analysis can be found in [93], and are not provided here. Instead, we summarize the adjoint sensitivity analysis for a linear circuit described by a set of linear equations $\mathbf{Mx} = \mathbf{b}$. By neglecting second-order variations, we have:

$$\delta\mathbf{x} = \mathbf{M}^{-1}[\delta\mathbf{b} - \delta\mathbf{M}\mathbf{x}] \tag{2.7}$$

For a scalar performance function $f(\mathbf{x})$, we can write

$$\delta f = \left[\frac{\partial f}{\partial \mathbf{x}}\right]^T \delta\mathbf{x} = \left[\frac{\partial f}{\partial \mathbf{x}}\right]^T \mathbf{M}^{-1}[\delta\mathbf{b} - \delta\mathbf{M}\mathbf{x}] \tag{2.8}$$

Postulating a vector ξ such that $\mathbf{M}^T\xi = \left[\frac{\partial f}{\partial \mathbf{x}}\right]$, we can have the following equation by substituting ξ into equation (2.8):

$$\delta f = \xi^T[\delta\mathbf{b} - \delta\mathbf{M}\mathbf{x}] \tag{2.9}$$

Using this expression, the sensitivities of f with respect to any entry of matrix $\mathbf{M}$ or vector $\mathbf{b}$ can be easily computed once ξ is known. The advantage of adjoint sensitivity analysis is that, ξ needs to be calculated only once even if we want to obtain multiple sensitivities. Furthermore, ξ can be easily calculated with the LU factors of $\mathbf{M}$, which could have been obtained in solving $\mathbf{Mx} = \mathbf{b}$.

2.5 Rent's Rule

Rent's rule, first revealed in [74], is a statistical predictor of the average number of terminals that a group of modules in a circuit requires for communication with the rest of the system, It bears the simple form:

$$T = kN^p \tag{2.10}$$

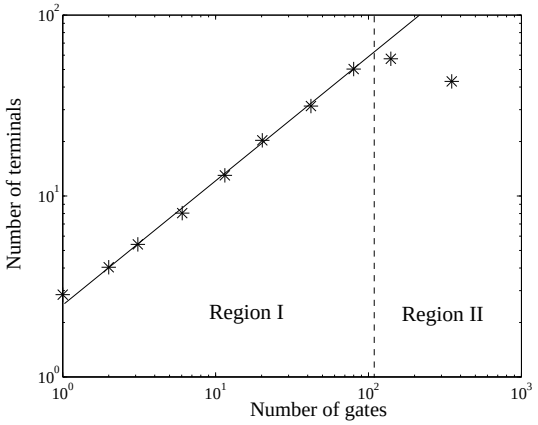


Figure 2.10: Terminal-gate relationship for a hierarchical bi-partitioning of a benchmark circuit (ISCAS 's953') [26].

where T is the number of signal input and output (I/O) terminals of the group of modules, N is the number of gates in the group, k is the Rent's coefficient, interpreted as the average number of terminals for a single gate, and p is the Rent's exponent. In practice, Rent's parameters p and k for a circuit are typically extracted through a circuit partitioning process. Figure 2.10 shows typical Rent data extracted from a benchmark (after [26]). The terminal-gate relationship was obtained using a recursive bi-partitioning process; for each level, the coordinates of the data point (represented as "$*$" in Figure 2.10) take the average number of terminals and gates over all partitioned groups at that level. As shown in the figure, the majority of the data fits accurately with the exponential relationship (2.10), and this region of the figure is labeled as "Region I". However, as the number of gates N exceeds some critical size, the relationship between T and N deviates from the exponential curve and T can keep constant or drop while N increases; this is first observed in [74]. The right part of Figure 2.10 shows an example of such deviation, and the number of terminals runs low and eventually decreases with a growing gate number; we label this region as "Region II" after [74]. The existence of "Region II" lies on the fact that the number of terminals becomes constrained by the limited number of input/output terminals at the chip periphery as the number of gates approaches the total number of gates on the chip.

Rent's parameters p and k can give an indication of the interconnection complexity of a circuit [9]. In particular, a small variation of p, given its position as an exponent, can have dispro-

portionate effects on secondary circuit properties such as the wire length. Based on Rent's rule, various system level prediction techniques have been developed to estimate and predict important circuit properties, such as wire length distribution [38, 41, 106], fanout distribution [120] and buffer distribution [123]. In this thesis, Rent's rule is employed to estimate and guide the distribution of buffers in a new family of circuits, structured ASICs.

Chapter 3

Crosstalk Noise Reduction in Global Routing

As technologies scale down, interconnect performance is greatly affected by crosstalk noise due to the decreasing wire separation and increased wire aspect ratio, and crosstalk has become a major bottleneck for design closure. The effectiveness of traditional buffering and spacing techniques for noise reduction is constrained by the limited available resources on chip. In this chapter, we present a method for incorporating crosstalk reduction criteria into global routing under a broad power supply network paradigm. This method utilizes power/ground wires as shields between signal wires to reduce capacitive coupling, while considering the constraints imposed by limited routing and buffering resources. An iterative procedure is employed to route signal wires, assign supply shields, and insert buffers so that both buffer/routing capacity and signal integrity goals are met. In each iteration, shield assignment and buffer insertion are considered simultaneously via a dynamic programming-like approach. Our noise calculations are based on Devgan's metric, and our work demonstrates, for the first time, that this metric shows good fidelity. An effective noise margin inflation technique is also proposed to compensate for the pessimism of Devgan's metric. Experimental results on testcases with up to about 10,000 nets point towards an asymptotic run time that increases linearly with the number of nets. Our algorithm achieves noise reduction

improvements of up to 53% and 28%, respectively, compared to methods considering only buffer insertion, or only shield insertion after buffer planning.

3.1 Introduction

As VLSI design advances into the nanometer regime, besides optimizing interconnect timing and power for performance improvement, it is also important to integrate the analysis and optimization of interconnect crosstalk noise in order to maintain signal integrity. With each successive technology generation, wires are spaced closer to each other and also have more skewed aspect ratios, resulting in increased levels of coupling capacitance in each generation, which can cause a switching net to induce large noise spikes on its neighboring nets. Crosstalk noise can affect the circuit performance in one of the two ways:

(a) *Functional noise* is seen when a victim net changes its level due to the switching of its neighbor aggressor nets, and this could lead to circuit malfunction.

(b) *Delay noise* is caused when the victim and aggressor nets switch at the same time, causing the injection of a noise pulse during switching, which can alter the delay.

The algorithm proposed in this chapter primarily mitigates functional noise; however, the insertion of buffer and supply shields with stable voltage levels between signal wires also provides the subsidiary benefit of greatly easing delay uncertainties and therefore relieves the delay noise.

Various noise analysis techniques have been proposed over the years. Crosstalk noise can be mostly accurately evaluated by circuit simulation methods [85], but this is computationally expensive for large circuits. Under a linear circuit model, noise analysis can be performed using model order reduction techniques as in [88, 92], through which noise waveforms can be obtained with high accuracy, although the computational cost can still be too high for its use in a full chip noise optimization tool. To further expedite noise analysis, various simplified crosstalk models have been proposed. [111] derives bounds for crosstalk noise using a lumped model but ignoring interconnect resistance and assuming a step input for aggressor driver. This effort is further extended

by [14, 32, 63, 112] to incorporate π model, 2-π model and 4-π model for interconnect modeling to take into account coupling location at victim net and better model aggressor net. Although these approaches can provide closed-form analytical model for noise attributes, their simplified coupling model fails to capture the distributed nature of coupling between adjacent lines; at the same time, the closed-form expressions are too complicated to be incorporated in a optimization engine in physical design. As extensions to Devgan's metric, whose model and virtues are detailed described in preliminary Section 2.4.2, [72] employs moment-matching technique to accurately estimate noise value in a RLC network, while [58] introduces a time constant to control an exponential noise decay at victim node to reduce pessimism. However, both approaches require multiple tree traversals to obtain an noise estimation, hence not easy to be embedded into noise optimization process. Most simplified, some existing noise optimizations in physical design employ a geometric noise metric [117, 128], in which the noise level is proportional to the overlap length between wires. However, this model is not accurate since it does not capture the electrical properties of circuit.

Based on the above review and comparison, we employ Devgan's metric, among other noise analysis models, in our noise reduction algorithm during global routing. This is mainly due to the fact that Devgan's metric considers a wide range of electrical properties and bears a extremely concise form as discussed in Section 2.4.2. Furthermore, this chapter studies the fidelity of Devgan's metric and proposes a noise margin inflation technique to overcome its pessimism at high slew rates [58, 72] so as to to safely employ this metric. Therefore, Devgan's metric fits well into our global routing flow for noise reduction.

The efforts for crosstalk noise mitigation are exerted throughout physical design stages. In early design stages, [22, 77] address the problem of buffer planning for better noise reduction during floorplanning. Due to the lack of detailed physical information at this early stage, the optimization efforts can be quite limited. On the other hand, after the routing phase, with more detailed physical and electrical properties of a circuit known, crosstalk noise can be effectively reduced with various techniques. Many recent works [13, 52, 54, 101] present gate sizing methods to reduce noise by appropriately increasing the driving gate size of victim net and decreasing that

of aggressor net. Wire spacing, which can control the separation between nets, are utilized to optimize crosstalk noise in [82, 83]. Similarly, wire sizing is employed in [53] for noise reduction. A disadvantage of the above techniques is that they are all performed after the routing phase. As the routes and gates position are rather fixed, the solution space can be constrained, hence greatly limits the optimization efforts.

Among other physical design stages, routing determines the routes, layers and relative positions of nets, which are some of the most contributing factors to affect the noise level of a net; at the same time, some effective noise mitigation resources such as buffers and shields, can also be employed during routing. Therefore, noise reduction efforts *during* the routing phase can significantly improve the signal integrity of a VLSI circuit. Previous routing techniques [66, 69, 117, 128] use simple crosstalk metrics and/or ignore issues related to routing congestion, particularly due to supply nets and shields. A practical crosstalk-conscious router must consider the trade-off between routing resource consumption and noise reduction. To address this issue, shield planning and crosstalk budgeting are utilized in some recent works [62, 115, 116] to perform global routing with RLC noise avoidance and routing area optimization. However, buffer insertion, an effective way to block noise propagation, is not addressed in these works. [4] proposes an algorithm to utilize buffers for improved timing and noise performance, but this technique is performed in the post-layout phase, therefore suffers from constrained solution space. Moreover, it does not fully address the contention for buffer resources. With trends showing the use of an increasingly large number of buffers to reduce interconnect delay and achieve timing closure in modern designs, it is projected at 32nm technology, a very large proportion of all cells need to be buffers [98]. This will produce high levels of contention for limited buffer resources, implying that buffer considerations must be taken into account during global routing, for best resource utilization. This motivates our simultaneous buffer and shield insertion scheme for functional noise reduction. Together, shielding wires and buffers can effectively control crosstalk noise, and an integrated approach can manage both resources optimally.

This chapter addresses the problem of crosstalk noise reduction during global routing under restrictions on the availability of routing and buffer resources. Some of the contributions in this chapter are as follows:

- We simultaneously allocate power supply wires as shields and insert buffers in global routing phase in order to route nets under a noise budget, so as to best utilize noise avoidance resource.
- We develop a practical methodology to tackle the pessimism of Devgan's noise metric. For the first time, we verify the fidelity of this metric, and develop a noise margin inflation technique to compensate for the over-estimation of Devgan's metric.
- To utilize existing power supply wires, this method is presented under the backdrop of a flexible supply network architecture. The procedure results in a signal/power co-routing solution at the global level.
- We also incorporate considerations to insert a sufficient number of buffers to control the delays and slews on each signal line.
- Experiments show our algorithm achieves noise reduction improvements of up to 53% and 28%, respectively, compared to methods considering only buffer insertion, or only shield insertion after buffer planning; while testcases with up to about 10,000 nets point towards an asymptotic run time that increases linearly with the number of nets.

Our method works iteratively: starting with an initial global routing solution, an enumerative dynamic programming-like algorithm is used to simultaneously assign supply shields and buffers to meet the noise budget for each net, one at a time, to find a minimum cost solution for the net. Next, an iterative rip-up-and-reroute step is performed to better meet the routing and noise goal. We simultaneously take into account the limitations on routing/buffer resources and the needs for signal integrity and provide a global routing solution that is immune to capacitive coupling noise. For comparison purposes, we also implement an intelligent greedy approach which is faster, but less effective in resource allocation.

The organization of the chapter is as follows. In Section 3.2, the routing, noise and power grid models are described. We then formally present the problem formulation in Section 3.3. This is followed by the verification of the validity of using Devgan's metric in Section 3.4. Next, in Section 3.5, we describe our iterative algorithm. Experimental results are presented in Section 3.6, followed by a conclusion in Section 3.7.

3.2 Background

3.2.1 Global Routing and Buffer Model

Our global routing problem considers simultaneously routing a set of m nets, $\mathcal{N} = \{n_1, n_2, \ldots, n_m\}$. During the routing process, we build the global routing grid and global routing graph as described in Section 2.2.1 and find the routing path in terms of routing graph edges for each net. As power wires are considered at the same time with signal wire routing and will share routing resources, the routability requirement for a routing edge e can be more specified as: $S_e + P_e \leq C_e$, where S_e and P_e are the total widths (including wire spacing) used by signal and power lines passing edge e, and C_e is the geometrical width of the routing cell boundary associated with edge e; violation of this will result in *routing overflow*. In routing layer arrangement, we follow a routing model in which horizontal and vertical lines are routed on different layers so that each layer is dedicated to routing in one direction. The aim of routing is to eliminate routing overflow while achieving other performance-related goals such as noise reduction.

To effectively improve timing performance and block noise propagation, buffer insertion is addressed during the routing process. We employ the distributed buffer model as described in Section 2.3, and assume there are buffer sites pre-distributed across the layout plane. Applying routing grid to the layout plane as the tessellation discussed in Section 2.3, the number of buffer sites in each global routing cell (GRC) can be found, and they can be selectively implemented to be buffers and inserted to global nets passing the GRC. To emphasize the buffer planning role of routing grid, we redraw the global routing grid and its corresponding routing graph (which are

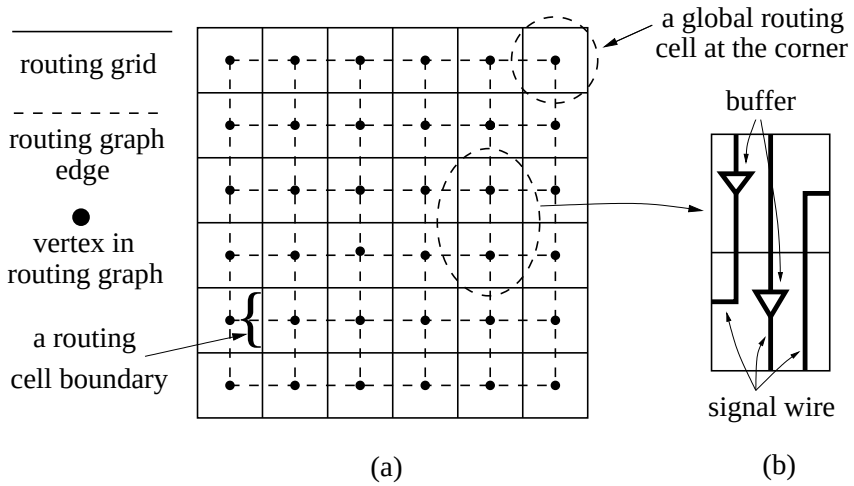


Figure 3.1: (a) A routing grid and the corresponding global routing graph. (b) Buffer sites in GRCs are implemented to be buffers which are inserted to signal wires.

discussed in Section 2.2.1) in Figure 3.1(a), and Figure 3.1(b) gives an inset view that buffer sites distributed in two neighboring GRCs are implemented to be buffers and inserted to globally routed nets. To control the interconnect delay and slew rate, a maximum wire length driven by a buffer (gate) is enforced as described in Section 2.3; in terms of global routing grid, this translates to a requirement that the maximum total interconnect length that can be driven by a buffer (gate) is the length of M GRCs.

3.2.2 Power Supply Architecture

A traditional power supply architecture is composed of a regular dense grid that traverses the entire layout area. However, different parts of the layout require various amounts of current, and the density of the grid does not have to be uniform. This flexibility is exploited in [103] to design a locally uniform, globally non-uniform power grid structure, and this property is also employed in [107] to optimize routability while achieving satisfiable power integrity.

In this work, we utilize a similar non-uniform and flexible power grid architecture that can satisfy the power requirements and provide shielding functionality between neighboring signal wires. We assume that the power grid is an array of variable density, and the power integrity of the supply grid is maintained by ensuring that the average and minimum number of wires feeding

every block exceeds a specified threshold. The layout is divided into blocks, which corresponds to different functional blocks in practice. For each block i, regarding the power integrity, we are given:

- a Minimum Average Number (MAN) of supply wires MAN_i per grid edge.
- a Minimum Number (MN) of supply wires MN_i over each grid edge belonging to the block i.

Note that MAN_i and MN_i are both defined *per edge* of the routing graph of Section 3.2.1 and together define a basic structure of power network, thus enabling the power considerations to be incorporated even before a detailed power architecture is determined. Any extra power lines/shields beyond MN_i and MAN_i will only improve the power grid performance [107]. This power grid model works well on intermediate metal layers, as in [97], where the variable number of power lines in adjacent blocks do not have to match exactly since they can be connected to each other through upper layers. The edge capacities in the routing graph are shared by signal wires and power supply wires, implying that if signal wires utilize too much of the routing capacity at a boundary, then it is not possible to make enough room for supply wires. The supply wires in the supply grid are used not only to carry power currents, but also work as shields between aggressor and victim signal wires to reduce noise; we will use the terms *supply wire* and *shield* interchangeably.

3.2.3 Noise Calculation under Shield Insertion

Supply Shield Arrangement

The insertion of a supply wire between two signal wires will shield them from each other. As supply wire has a stable voltage level, it can effectively mitigate the capacitive coupling noise between the signal lines. We say that a side of a signal wire is provided *protection* if it neighbors a supply shield. Figure 3.2 shows five aggressor nets and a two-sink victim net with two supply wires as shields. As shown in the figure, with the insertion of shield s_1 and s_2, aggressor A_5 will not affect the victim net, and aggressor A_4 will also have less capacitive coupling with the victim

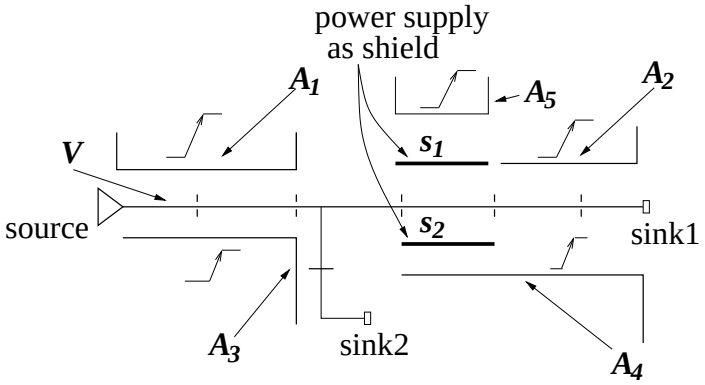


Figure 3.2: Switching noise on a victim net with shielding supply wires. V is the victim net, A_i, $i = 1, \ldots, 5$ are potential aggressor nets, and s_1, s_2 are power supply shields.

net compared with the case when s_2 is absent.

In the global routing phase, the exact positions of signal nets are still undetermined, and hence neighborhood information is not fully available. At the same time, the switching activity of nearby signal nets is also unclear. In the absence of this detailed information, it is meaningful to determine a *worst-case* scenario based on the information that is available. We assume if one side of a signal net is not placed next to a shield, it will (pessimistically) be adjacent to an aggressor net that will induce coupling noise on the net. Thus a signal wire must have supply lines placed on both sides to be fully protected. However, due to limited routing resources, a net may not be fully protected. We refer to the number of protected sides of net k at a specific routing edge e_{ij} as $P_{ij,\ k}$, and it can take a value of 0, 1, or 2. If $P_{ij,\ k}$ is known for the signal wires across edge e_{ij}, the number of supply wires required for shielding can be found as follows. If there are S signal wires requiring protection on a single side, and D signal wires requiring protection on both sides, we must have p power supply as shields to achieve the protection, and the minimum amount of p can be found by the following equation:

$$p = \begin{cases} \frac{S}{2} + D & \text{if } S \text{ is even} \\ \frac{S+1}{2} + D & \text{if } S \text{ is odd} \end{cases} \tag{3.1}$$

It is easy to prove that an arrangement of supply and signal wires with the above numbers exists, so that the desired protection is feasible. The specific positions of the signal wires and supply wires

will be handled by detailed routing tools. Since supply wires and signal wires share the routing resources, the capacity constraint of edge e_{ij} requires that

$$C_e \geq w_s \cdot s_e + w_p \cdot p_e \tag{3.2}$$

where C_e is the boundary capacity; w_s and w_p are the width (including the metal width and the spacing) of a signal wire and a supply wire, respectively; s_e and p_e are the number of signal wires and supply wires passing the boundary e, respectively.

Noise Calculation with Devgan's Metric

We assume that a *noise margin*, NM_{spec}, is specified for each gate or buffer input in the circuit, and represents the largest noise voltage that will avoid a circuit malfunction[1]. An net is represented using a standard segmented coupled RC model so that its equivalent circuit contains a tree of resistors with capacitors to ground and to adjacent nets. The nodes in the tree are assigned parental relationships, with nodes closer to the source preceding those away from it. Our noise reduction algorithm propagates a set of candidate solutions in the search space, and to facilitate this, we recursively define the noise margin for an internal node i as:

$$NM(i) = \min_{all\ child\ nodes\ j}(NM(j) - V_n(i \leftrightarrow j)), \tag{3.3}$$

where $V_n(i \leftrightarrow j)$ is the noise voltage induced between i and j. A net is *noise free* if at any driving node (i.e., the net source or any buffer output), we have:

$$I_n \cdot R_d \leq NM \tag{3.4}$$

Here, R_d is the gate driver resistance, and I_n is the induced noise. The noise current can be calculated and propagated using Devgan's metric as described in Section 2.4.2. This is illustrated

[1]As described later in Section 3.4.2, this noise margin can be selected by inflating the actually desired noise margin, so that it accounts for the over-estimation in Devgan's metric.

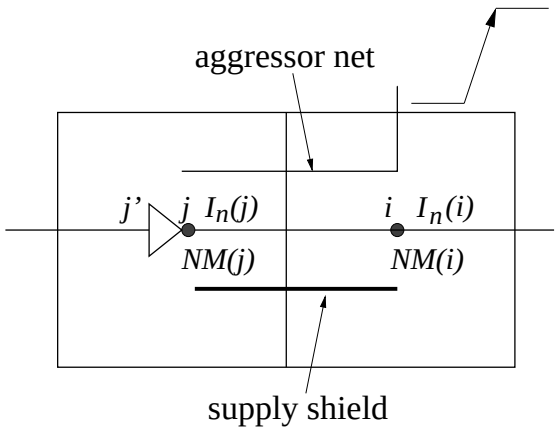


Figure 3.3: Calculation of noise margin by Devgan's metric.

by an example in Figure 3.3, which shows a signal wire segment extending from the center of GRC i to the center of its neighbor GRC j. Let the per unit length coupling capacitance and resistance be C_c and R_e, respectively, and the aggressor voltage slew rate be μ. Since the lower side of the wire segment is shielded, only the upper aggressor will induce noise on this net, implying the following result according to Devgan's metric:

$$I_n(j) = I_n(i) + L_e \cdot C_c \cdot \mu \tag{3.5}$$

$$NM(j) = NM(i) - R_e L_e \left(\frac{1}{2} C_c L_e \mu + I_n(i) \right) \tag{3.6}$$

If the noise at j satisfies the constraint (3.4), the buffer will block the propagation of noise, and the noise margin at j' will recover to NM_{spec}, with the noise current $I_n(j')$ reset to 0 as well.

3.3 Problem Formulation

The formal statement of the problem is as follows. Given a tiling of a chip and the corresponding routing graph $G = (V, E)$, a set of nets $\mathcal{N} = \{n_1,\ n_2,\ ..., n_m\}$, the edge capacity C_{ij} for every edge $e_{ij} \in E$, and the buffer capacity B_i for each routing cell $i \in V$, the problem is to find a routing solution that:

1. determines the routes for each net on the routing graph,

2. satisfies the power wire density constraints, i.e., the average and minimum density MAN_i and MN_i of each block must be met,

3. determines the GRCs in which a net is to be buffered, subject to the buffer capacity constraint,

4. finds $P_{ij,\ k}$ over each edge e_{ij} in the routing of net k, subject to the edge capacity constraint (3.2),

5. satisfies noise constraint (3.4) for all nets.

6. ensures that the total amount of interconnects that can be driven by a buffer (gate) is at most M grid units.

3.4 Validity of Devgan's Metric

As discussed in Section 2.4.2, Devgan's metric is known to provide an upper bound for the crosstalk noise in an RC circuit, but this upper bound is also known to potentially result in large overestimates [58, 72]. We will now examine the utility of this metric, despite its known inaccuracies, in the specific context of our simultaneous shield and buffer insertion algorithm. In our approach, a set of candidate solutions is propagated, and the calculated noise is used to prune some of these solutions in one of the following two ways:

- Comparing the noise value V_{n,s_1} of solution s_1 and V_{n,s_2} of solution s_2 to identify the relatively larger one.

- Comparing the noise value $V_{n,s}$ of a solution s against a specified noise margin NM_{spec}, and pruning the solution if $V_{n,s} > NM_{spec}$.

We claim that for the above purposes, we can still safely use Devgan's metric despite its pessimism, and we will justify this by arguing for the fidelity of the metric, as well as the employment of a noise margin inflation technique, which can compensate for its pessimism.

3.4.1 Fidelity of Devgan's Metric

In the context of determining the relative order of protection solutions by their noise value, we argue that it is the *fidelity* of a noise metric that is important rather than the accuracy. Fidelity of a noise metric for protection solution refers to the degree to which an optimal or near-optimal solution according to this metric will also be near-optimal according to an accurate noise metric. During solution pruning, if the noise metric employed has good fidelity, the relative order determined by this metric will have a high probability of being the same as that determined by an accurate simulation method, such as SPICE. Thus we can claim that solutions pruned under the noise metric are truly inferior to others in terms of noise level.

We verify the *fidelity* of Devgan's metric with a set of experiments that are conducted under similar design scenarios as those encountered for our problem. We randomly generate the pin locations of a circuit with several multiple-sink nets in a 6×6 grid, and then route the nets using AHHK algorithm [6]. The coupling capacitances are then extracted using the technology parameters that will be detailed later in Section 3.6. Next, we randomly pick one net as the victim net and consider the other nets as aggressors, and assume all aggressor nets adversely switch at the same time while the victim net remains at a stable value. Under this scenario, we compute the coupling noises at the sinks of the victim net using Devgan's metric and using a SPICE simulation. The above experiment is repeated 100 times, and then all of the victim sinks in these experiments are ranked according to their noise under Devgan's metric and under SPICE, respectively. Since a net has an average of 2.5 sinks under our experiments, the noise is computed at a total of about 250 sinks in these 100 experiments. The rank difference of each such sink under the two metrics is then determined.

Our experiments correspond to the following cases:

- The number of nets in each experiment is either 20 or 40: under these assumptions, we find that the average number of crosstalk adjacencies per victim net is around 10 grids and 13 grids, respectively, which indicates that there is a significant amount of coupling. The two cases correspond to low and high net density in a routing region.

- The input waveform, modeled as a saturated ramp from 0 to 1.8V, has a transition time of either 100 *p*s or 200 *p*s, which simulate different aggressor switching time. Here we assume a uniform transition time for all aggressor nets, which is consistent with the noise model used in our shield and buffer insertion algorithm.

This results in four combinations of (number of nets = 20 or 40) and (transition time = 100 *p*s or 200 *p*s), and the corresponding results that show the rank difference of each sink are listed in Table 3.1. For all 250 or so sinks in the experimental setup, the average errors that corresponds to the distance in ranking (between Devgan's metric and under SPICE) are around $9 \sim 12\%$, which suggests that Devgan's metric has good fidelity in comparing crosstalk noise. On average, while comparing two structures, if one of them has lower crosstalk noise than the other under Devgan's metric, it is very likely to also demonstrate lower noise under a SPICE simulation. The fidelity is even higher for the worst noise victim sink ranking for the 100 sinks with worst noise values in victim net, where the error is only around $4 \sim 11\%$.

These results lead to an important conclusion: *on average, Devgan's metric has good fidelity, and can be used as an effective metric to compare noise levels in our shield and buffer insertion algorithm.*

Aggressor rise time	# nets in circuit	Ranking: All victim sinks			Ranking: Worst noise sinks			average overlap (# of grid lengths)
		# victim sinks	Average rank diff.	% error	# worst noise sinks	Average rank diff.	% error	
200 *ps*	20	234	23	9.8%	100	6	6%	9.59
100 *ps*	20	264	32	12.1%	100	11	11%	10.49
200 *ps*	40	279	16	9.3%	100	4	4%	13.37
100 *ps*	40	271	33	11.1%	100	8	8%	13.61

Table 3.1: Verification of the fidelity of Devgan's metric.

3.4.2 Noise Margin Inflation

The second pruning technique in our algorithm compares the noise value estimated from Devgan's metric against a specified noise margin NM_{spec}. In this comparison, due to the pessimistic nature of Devgan's metric, a noise value can be over-estimated, and thus the corresponding eligible solution can be falsely pruned, so that our buffer and shield insertion algorithm may over-optimize

and use more than enough resources to accomplish full protection, or under stringent protection resources, may result in false-failures. To compensate for this pessimism of Devgan's metric, in practice, we can apply a noise margin inflation technique, i.e., we can heuristically inflate the specified noise margin NM_{spec} to be NM_{inf} ($NM_{inf} > NM_{spec}$). If chosen carefully, the inflated noise margin as input to our algorithm will compensate for the over-estimation of Devgan's metric.

The inflated noise margin NM_{inf} can be estimated from NM_{spec} using a curve-fitting technique. If we denote a noise voltage evaluated under Devgan's noise metric as V_d, and the value of the same noise evaluated under a SPICE simulation (which is considered as the accurate noise level) as V_s, then we use a fitting function $V_d \approx a_0 V_s + a_1$, where a_0 and a_1 are constants. Therefore, the inflated noise margin can be obtained from the relationship as $NM_{inf} = a_0 NM_{spec} + a_1$.

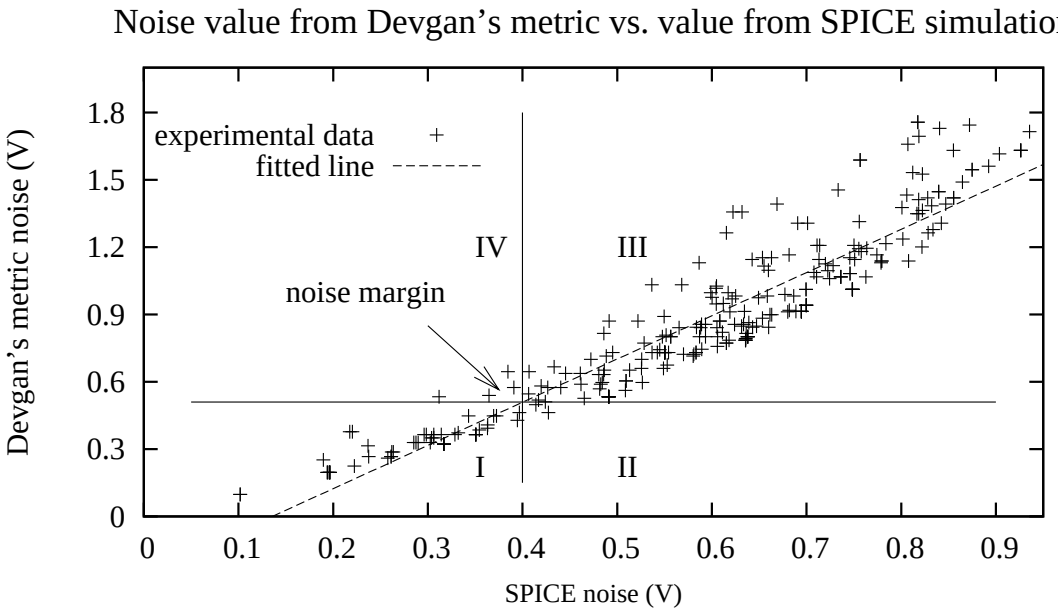


Figure 3.4: Noise estimation under Devgan's metric and SPICE simulation. The least squares fit to a linear approximation between the two noise metrics is drawn as a dashed line. The plane is divided into four representative regions, I through IV, as shown above.

To generate the coefficients a_0 and a_1, we first randomly generate nets in a circuit with the same technical parameters as in Section 3.4.1. The number of nets in each circuit is 30 to simulate the congested routing scenario of a complex design, and the aggressor transition time is 200 ps, which is consistent with our experimental setup in Section 3.6. The experiments are performed

100 times, and the noise value at each sink is evaluated with both Devgan's metric (obtained as V_d) and SPICE simulation (obtained as V_s). We plot the distribution of V_d vs. V_s as in Figure 3.4, and each point in the figure has a coordinate of (V_s, V_d) which corresponds to noise value evaluated under different metrics. A least squares fit is then performed to find the values a_0 and a_1, and the result is shown by the dashed line in the figure.

It is noted that we eliminate all the noise points with $V_d > V_{dd} = 1.8$V in performing the curve fitting for the following reasons: (a) It is not physically possible for a noise value to be larger than V_{dd} in a RC interconnect model, and Devgan's metric is known to be over-pessimistic here. (b) In practice, we observe that for noise with $V_d > V_{dd}$, the corresponding V_s is always much larger than a practically specified noise margin, and this eliminates the necessity to include them in the sample data. From the fitting results, we can obtain the values $a_0 = 1.93$ and $a_1 = -0.26$V under our experimental setup.

To illustrate the usefulness of the noise margin inflation technique, we draw a vertical line at a given noise margin point, and a horizontal line at the point where this vertical line crosses the fitting line. These horizontal and vertical lines divide the plane into four regions labeled as I, II, III and IV. We will use N_i, i = I, II, III, IV to indicate the number of points falling in each region i. Figure 3.4 shows an example of such crossing lines and the corresponding divided regions under a specified noise margin $NM_{spec} = 0.4$V. Points falling in region I [III] represent the cases in which a noise satisfies [violates] both the NM_{spce} under SPICE simulation and NM_{inf} under Devgan's metric. As shown in Figure 3.4, most of the points fall in regions I and III, which indicates that under our noise margin inflation technique, the qualitative noise value from Devgan's metric agrees well with the SPICE simulation result. For the noise margin inflation to be an effective technique, we desire that it should capture all the valid solutions, and thus a valid solution under SPICE simulation will not be falsely discarded under inflated noise margin. This effectiveness is reflected by the very small number of points in region IV, which are the points violating NM_{inf} under Devgan's metric but actually satisfying NM_{spec} under SPICE simulation. In the above example, only 1.7% (calculated as $N_{IV}/(N_{III} + N_{IV})$) of the total discarded noise values with

our technique are actually good solutions but falsely pruned. Interpreted alternatively, our noise inflation technique can capture 96.4% (calculated as $N_I/(N_I+N_{IV})$) of all the eligible solutions in a statistical sense. At the same time, for noise inflation to be a safe technique, we also want to avoid false acceptance of a noise-failure solution, i.e., the protection solution obtained under the inflated noise margin NM_{inf} should also satisfy the original noise margin requirement NM_{spec}. This is validated observing that a very small number of points fall into region II in Figure 3.4. This safety requirement is crucial for the accuracy of our method, and we will further experimentally show in Section 3.6 that the noise margin inflation technique embedded in our simultaneous shield and buffer insertion algorithm does generate valid noise protection solutions, and the above proposed curve-fitting technique to estimate inflated noise margin is an accurate approach in practice.

3.5 Routing and Crosstalk Reduction Algorithm

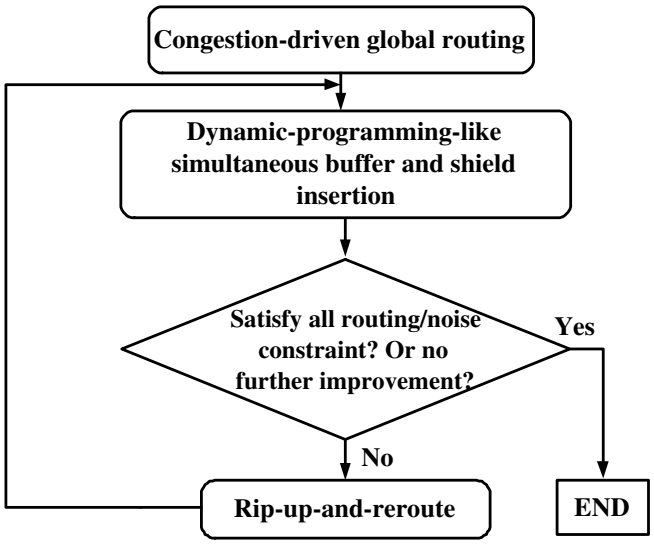


Figure 3.5: Overall flow of global routing with simultaneous shield and buffer insertion for crosstalk noise reduction.

Our approach to the problem formulated in Section 3.3 is depicted in Figure 3.5. The algorithm is iterative and proceeds through three steps: congestion-driven global routing, a dynamic programming-like simultaneous buffer and shield insertion procedure and rip-up-and-reroute refinements. Steps 2 and 3 iterate until all constraints are satisfied, or no further improvement is

possible. In the above buffer and shield insertion and rip-up-and-reroute, we process one net at a time and maintain a fixed order of all nets. We have experimentally found under different randomly chosen net orderings, the results change very little, as long as we maintain the same fixed net order through all of the iterations. This is due to the fact that the early iterations are seen to create good estimates of resource utilization, and this reduces the order dependence.

3.5.1 Step 1: Congestion Driven Routing

The signal wire routing procedure consists of two phases, similar to that in [5]. The first phase of routing constructs Steiner trees using the AHHK algorithm [6], and works as a fast estimator of the congestion map. The second phase performs a congestion-driven rip-up-and-reroute based on this initial solution, with the objective of minimizing the congestion cost over routing grid edges [5]. If there is still an overflow violation after this phase, more rip-up-and-reroute steps will be performed (with the same net order), as in [86].

Besides punishing regular congestion overflow, the congestion cost function in the congestion-driven routing is designed to also incorporate the consideration of power supply requirements. Since all of the routing capacity is shared by signal and power routing, signal routing results will determine the power supply structure, and thus signal routing must leave enough capacity for power supply to satisfy the average and minimum power supply densities MAN_i and MN_i for a block i of the routing region. The cost function for a signal wire traversing an edge e in block i is composed of two terms:

$$\text{routing cost} = cost_e + cost_i \tag{3.7}$$

in which $cost_e$ is the cost of traversing edge e, and penalizes any violation of MN_i for edge e in block i; $cost_i$ is the cost of passing block i, and penalizes any violation of MAN_i of block i. Both

terms take the following form:

$$cost = \begin{cases} \frac{U}{R} & \text{if } R > 0 \\ 10^{-R} & \text{if } R \leq 0 \end{cases} \tag{3.8}$$

where R is the residual signal routing capacity on the edge [block] for the first [second] term. This value is calculated by subtracting, from the total edge [block] capacity, the power supply requirements and the capacity already used for signal routing, U. Note for the first term, the power supply requirement is MN_i; for the second term, it is MAN_i multiplied by the number of edges in a block i, since MAN_i is defined *per edge*. The exponential form of the cost function after the capacity violation punishes the over-use of capacity from signal routing, so as the power grid requirements can be met. In addition, it can also effectively avoid the aggregation of signal wires.

The congestion driven routing performs a wave-form expansion starting from the source, and update tile cost with the lowest cost in each expansion like [6]. The routing of a net terminates upon all sinks have been reached. After every routing or rerouting, each of the routing nodes representing the GRC in the final path is added to the tree as an *Internal Node (IN)*, and a net will then be a set $\{s \cup P \cup IN \cup E\}$, where s is the source node, P is the set of sink pins, IN is the set of internal nodes, and E is the set of edges; this data structure is used for the procedure in Step 2, which follows this.

3.5.2 Step 2: Buffer and Shield Insertion

With a routing solution from the above step, we simultaneously allocate shield and buffer resources to each net so that the solution can meet the noise requirement while using least protection resources. Each net is processed individually, and the procedure traverses the tree structure of the net in a bottom-up manner, starting from the sinks and moving towards the source, moving along one grid square at a time. Each tree is described in terms of nodes that correspond to the GRCs that it passes through. Assigning a direction to the tree from the source to the sinks, we refer to the GRC that contains the immediate predecessor [successor] of a given node n in the tree

as the parent [child] cell of the GRC containing n. If a GRC has more than two children, we can insert pseudo-nodes, so that the final tree is a binary tree for ease of later processing.

While traversing a net across a GRC i, we have two methods for protecting it from crosstalk noise:

1. By deciding whether to insert a buffer or not at GRC i: this corresponds to two possible buffer insertion configurations (0 or 1 buffer).

2. By protecting the net using supply shields on one side, on both sides, or choosing not to shield the net at all. If GRC i is not the root of the tree, shield(s) may be inserted alongside the edge e_{ij} connecting current GRC i and its parent cell j in the tree. This results in three possible configurations (0, 1, or 2 sides protected).

Therefore, in each bottom up step, we may have six possible configurations for the *protection structure*. However, we cannot locally determine at each grid point which scheme is globally optimal, and therefore an enumerative dynamic programming-like approach is adopted here in the same spirit as in Van Ginneken's algorithm [110].

Protection Cost and Solution Architecture

While traversing a net bottom-up, at GRC i, we must find the protection cost corresponding to a protection structure, so as to measure the resource usage. For the protection cost related to buffer insertion, since the nets are processed one at a time, at any point in our insertion algorithm, the probability that an unprocessed net n_i crossing GRC i will insert a buffer from i is $1/M$. Let p_i be the sum of these probabilities over all unprocessed nets crossing cell i, and the cost for insertion of $b_{req}(= 0 \text{ or } 1)$ buffer at a specific cell i is similar to that in [5]:

$$cost_{buffer\ i}(b_{req}) = \begin{cases} 0 & \text{if } b_{req} = 0 \\ \frac{b_i + p_i + 1}{B_i - b_i} & \text{if } b_{req} = 1 \text{ and } b_i + b_{req} \leq B_i \\ \infty & \text{otherwise} \end{cases} \tag{3.9}$$

This cost function will significantly increase the cost penalty as buffer resources become contentious. For shielding cost, we can calculate the number of power supply wires required based on the number of sides to be protected N_{sh} and equation (3.1). In the same spirit to punish contentious resource usage, the shielding cost $cost_{shield\ ij}$ for a signal wire can be obtained from a similar form of equation as (3.9) above, but the predicted shield usage takes a different approach: each unprocessed signal net will probabilistically have 1 side to be protected (assuming equal probability for $N_{sh} = 0$, 1, or 2). Both the shielding cost and buffer cost are a measurement of the number of resources used, and they are approximately of the same order of magnitude. Therefore, we can combine them with a weighting factor λ (determined by resources availability) to develop a metric for resource usage, which we call the *protection cost* at GRC i, denoted as PC_i:

$$PC_i(b_{req}, N_{sh}) = \lambda cost_{buffer\ i}(b_{req}) + cost_{shield\ ij}(N_{sh})$$

where j is the parent cell of GRC i. This comprehensive cost function can be used as a metric to compare resource usages from different insertion schemes, and our goal is to find a minimum cost scheme satisfying the noise requirement, so as to resolve the contention for protection resources among nets.

At a cell i during the bottom-up traversal, the noise margin and noise current will vary according to the protection structure we choose, i.e., whether a buffer is inserted and the number of sides of the net that are shielded. If a buffer is inserted, the noise current will be "reset" to 0 and the noise margin set back to NM_{spec}. At each unbuffered location, depending on the number of sides of a signal wire that are shielded, we can have the following from equations (3.5) and (3.6):

$$\Delta I_n(N_{sh}) = (2 - N_{sh}) \cdot L_e \cdot C_c \cdot \mu \tag{3.10}$$

$$\Delta NM(N_{sh}) = R_e \cdot (\frac{1}{2} \cdot (2 - N_{sh}) L_e C_c \mu + I_n(i)) \tag{3.11}$$

where ΔI_n is the increase in the noise current due to the number of sides getting shielded $N_{sh} \in \{0, 1, 2\}$, and ΔNM is the noise margin decrease during the bottom-up step. To keep a record of

the protection structure in the enumeration, we define a *protection solution* at GRC i to be a 4-tuple $S = \{PC, NM, I_n, stru\}$, where NM is the noise margin at the end of the edge connecting GRC i to its parent cell j, I_n is the noise current induced by the neighboring signal wire at the same point, and PC is the protection cost of the current solution. The last component, *stru*=$\{$*buffer,* $N_{sh}\}$, represents the protection structure of the solution, where *buffer* is a binary number representing the number of buffers utilized at GRC i, and N_{sh} is the number of sides (0, 1 or 2) on which the wire is protected.

To satisfy the constraint that the total amount of interconnect can be driven by a buffer (gate) is at most the length of M GRCs, we maintain a *solution set array* (*SSA*) of length M at each GRC. Each element in *SSA* is a set of solutions, and the array is indexed from 0 to $M-1$. The solutions in $SSA[k], k \neq 0$ correspond to a total downstream interconnect of k GRCs to the nearest downstream buffer(s), and $SSA[0]$ stores the solution that correspond to the insertion of a buffer at the current GRC.

Algorithm for Building Protection Solutions

A unitary step in the enumeration algorithm is to build the solution set array at the current GRC based on the arrays at its child cell(s). Solutions in set $SSA[k]$, $k \neq 0$ are propagated from solutions in the lower indexed sets of the children GRCs. During the process, we update the noise margin according to different shielding choices to form new solutions, and the least cost child solution will be used to build solution in $SSA[0]$ in which buffer is inserted right at current GRC. The main procedure in the algorithm calls function `Find_sol_set_array` at the source cell, and returns the minimum cost solution that satisfies the noise constraint. The pseudo code for the algorithm is listed in Figure 3.6.

Function `Find_sol_set_array` finds the solution set array for a GRC. SSA is first initialized in line 1, and line 2 initializes the trivial case of a leaf cell , in which both noise current and protection cost are set to 0. Next, line 3 considers a single child cell and builds $SSA[i]$ from $SSA[i-1]$ of the child cell as illustrated in the left part of Figure 3.7. It considers the three

Algorithm: `Shield_buffer_insertion_for_noise_reduction`

Input: Net $N = \{s \cup P \cup IN \cup E\}$
Output: A protection solution with least protection cost at source s

1. SSA = `Find_sol_set_array`(s)
2. return protection solution $S = \{PC,\ NM,\ I_n,\ stru\} \in SSA$, with PC is minimized.

Function: `Find_sol_set_array`

Input: t is the GRC to be processed.
Output: The SSA of this GRC.

1. $SSA[i] = \Phi$, $i = 0, 1, ...M-1$
2. **if** $t \in P$, is a leaf
   for $i = 0$ to $M-1$
   $SSA[i] = SSA[i] \cup \{0, NM_{spec}, 0, \Phi\}$;
3. **else if** t has one child l
   SSA_l = `Find_sol_set_array`(l);
   for $i = 1$ to $M-1$
   for each $S_l \in SSA_l[i-1]$
   $SSA[i] = SSA[i] \cup$ `Propagate`(S_l, t);
4. Take minimum PC solution S_m from SSA_l,
   $SSA[0] = SSA[0] \cup$ `Propagate(Insert_buf` $(S_m, t), t)$;
5. **else if** t has two children l and r
   SSA_l=`Find_sol_set_array`(l); SSA_r=`Find_sol_set_array`(r);
   for $i = 2$ to $M-1$
   for each $S_l \in SSA_l[j]$, $S_r \in SSA_r[k]$ and j+k=$i-2$
   $SSA[i] = SSA[i] \cup$`Propagate(Merge`$(S_l, S_r), t)$;
6. Take min $S_{ml}.PC + S_{mr}.PC$ solutions S_{ml},S_{mr} from SSA_l, SSA_r;
   $SSA[0] = SSA[0] \cup$`Propagate(Insert_buf (Merge`$(S_l, S_r), t), t)$;
7. Take minimum PC solution S_{ml} from SSA_l;
   for $i = 1$ to $M-1$
   for each $S_r \in SSA_r[i-1]$
   $SSA[i] = SSA[i] \cup$`Propagate(Merge(Insert_buf` $(S_{ml},t),S_r),t)$;
8. Take minimum PC solution S_{mr} from SSA_r
   for $i = 1$ to $M-1$
   for each $S_l \in SSA_l[i-1]$
   $SSA[i] = SSA[i] \cup$`Propagate(Merge(`S_l,`Insert_buf` $(S_{mr},t)),\ t)$;
9. Prune solution set array SSA;

Functions:

`Propagate`(S, t) /* Extend solution by one grid length upward*/
if t is source, **return** S;
else return $\{S.PC + PC_t(0,N_{sh}), S.NM - \Delta NM(N_{sh}), S.I_n + \Delta I_n(N_{sh}), S.stru \cup \{0, N_{sh}\}\}$,
$N_{sh} = 0, 1, 2$;

`Insert_buf`(S, t) /* Insert a buffer to existing solution S */
return $\{S.PC + PC_t(1,0),\ NM_{spec}, 0,\ S.stru\ \cup\ \{1,0\}\}$;

`Merge`(S_l, S_r) /* Merge two solutions */
return $\{S_l.PC + S_r.PC, min(S_l.NM, S_r.NM), S_l.I_n + S_r.I_n, S_l.stru \cup S_r.stru\}$;

Figure 3.6: Algorithm for building protection solution.

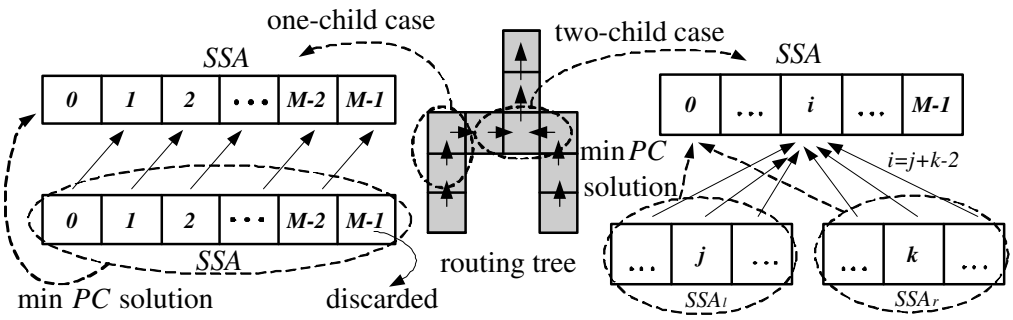


Figure 3.7: Updating SSA for one-child and two-child GRCs.

possibilities for the number of sides to be shielded: 0, 1 or 2. $SSA[M-1]$ of the child cell will be dropped to avoid violating the buffer load rule. In line 4, the protection solution with minimum protection cost is selected from the children solutions and a buffer is inserted. As a result, both the noise margin and noise current are recovered for this tuple. Line 5 deals with finding a solution set array for a GRC with two children. The complexity involved here is that when we build the solution, all of the combinations must be considered, as the right part of Figure 3.7 suggests. Line 6 picks the least cost solution from both the left and right children, and inserts a buffer to drive both children, thereby recovering the noise margin and noise current. However, a buffer can also be inserted only to the left or right branch. Lines 7 and 8 build solution tuples that correspond to these two cases, where a buffer is inserted to only one child branch and combined with solutions from the other. In the above steps, functions `Propagate`, `Insert_buf` and `Merge` are called to build solutions. There can be a large number of solutions in the SSA for a GRC after the `Propagate` and `Merge` steps; however, it is not necessary to keep all of them, and the solution pruning in the last steps will greatly reduce the solution set size. There are three types of pruning techniques employed here:

- We prune those solutions that violate the noise constraint (3.4).
- We prune the solutions that violate the buffer or wiring capacity.
- If a solution S_1 is provably *inferior* to another solution S_2 in the same SSA of a GRC, i.e., if $PC_1 > PC_2$, $NM_1 < NM_2$ and $I_{n1} > I_{n2}$, then it is pruned from the set of solutions.

While there is no concrete way of proving that the size of SSA will be small, the pruning techniques work efficiently in practice; our experiments show that the number of solutions at each GRC is limited between 3 and 60, and is less than 15 in most cases.

3.5.3 Step 3: Refinement

After the simultaneous shield and buffer insertion for noise reduction, refinement steps are applied if there are still some nets cannot be protected from noise constraint violation. The procedure is similar to that used in global routing phase. We rip-up and reroute all of the nets in the same fixed order as before. After one net is ripped up, it is rerouted immediately by the rerouting algorithm described in Section 3.5.1. However, the cost function in rerouting is now the combination of both the wiring congestion cost and the buffer congestion cost. This will drive the net to go through regions where wiring capacity and buffers are abundant. The dynamic programming-like algorithm for simultaneous shield and buffer insertion is then applied. After all of the nets have been ripped up, rerouted, and then protected, the whole refinement step will be performed again if there is still some noise violation. The iteration stops until noise violations are fixed or there is no further improvement. In practice, the algorithm will not iterate many times, and our experimental results shown that there will not be much improvement after the third iteration. To provide additional protection from noise, in the last step, the unprotected nets will greedily take all of the unused wiring and buffer capacities along its path; however, this step is optional.

3.5.4 Algorithm Complexity

The algorithm complexity mainly consists of two parts: the complexity of the routing and rerouting algorithm, and the complexity of the simultaneous shield and buffer insertion algorithm. In practice, these two parts of the algorithm only iterate a limited number of times before stop, and therefore the number of iterations will only contribute a constant factor to the overall complexity. The routing and rerouting process essentially follows the maze routing algorithm, and thus has a

complexity of $O(NGlogG)$, in which G is the total number of GRCs in the routing graph and N is the number of nets in the circuits. The computational complexity of our algorithm is dominated by the simultaneous shield and buffer insertion algorithm. For a single sink net, as function `Propagate` in Figure 3.6 suggests, the number of solutions will increase by 3 times for every bottom-up propagation. There are M solution sets, and all of the solutions in solution set $SSA[i]$ are actually derived from $SSA[i-1]$ of the child cell, with $SSA[M-1]$ of the child cell will be dropped. So the maximum number of solutions in a solution set can be 3^{M-1}, which is a constant. However, if there are $d > 1$ sinks in the tree, as line 5 in function `Find_sol_set_array` suggests, the maximum number of solutions can be as many as $O(C^d)$, where C is a constant. However, this is found to be unduly pessimistic in practice, since a large number of propagated solutions will be discarded as the pruning techniques works well. This also indicates why the approach is called "dynamic programming-like" instead of "dynamic programming": we have no provable optimal substructure property. However, our experiments show that the pruning method is efficient and the number of unpruned solutions at each node is small, as mentioned at the end of Section 3.5.2. Thus the running time for processing one net will not explode exponentially; instead, it will be asymptotically linear with the net length. Experimentally, we also observe that the asymptotic total run time increases linearly with the number of nets in the benchmark circuits.

3.6 Experimental Results

3.6.1 Experimental comparisons

Our algorithm is implemented in C++ on a Linux PC with a 2.8GHz CPU and 1GB memory. Out of the 16 benchmarks, the first 14 benchmarks in Table 3.2 are 8 benchmarks obtained from the authors of [5] and 6 GSRC benchmarks (*n30c, n50c, n100a, n200a, n200b, n300*) [36]. The largest benchmarks *syn1* and *syn2* with over 10,000 nets are randomly generated[2]. We superimpose a routing grid over the the floorplan so that the geometry of each GRC is almost a square. We

[2]We did not use the ISPD98 placement benchmarks, because most of the nets in it are very short, which makes buffer insertion unnecessary, and cannot be used to illustrate the buffer contention problems that are projected for future technology nodes. This is consistent with the experience of the authors of [31].

assume that layer assignment has already been performed for the wires, and our router handles each layer separately. For simplicity, our results show one horizontal and one vertical layer, but the approach is easily extended to multiple layers. The number of buffer sites in each GRC is generated randomly and the total number of available buffers is listed in Table 3.2. We divide the design into several blocks, which correspond to different styles of circuits, such as control logic, data path, etc., and in our experiments, we use 7 blocks. The power supply requirements MAN_i and MN_i are randomly generated but in a balanced manner across the chip (in practice, these will be dictated by the functional blocks). We also assume that a power grid wire is twice the width of a signal wire. The routing edge capacities are assigned as shown in the table, in the units of signal wire width. We assume that the grid length $L_e = 600\mu$m, that for all gates, the noise margin specification $NM_{spec} = 0.4$V under a V_{dd} of 1.8V, and that the aggressor voltage change rate $\mu = 9 \times 10^9$V/s. The technology parameters used in the experiments are derived from [27] and [7] for the 0.18 μm technology: unit length coupling capacitance C_c=0.0583fF/μm, unit length resistance R_e=0.373Ω/μm, and buffer driver resistance R_d=180Ω.

Circuit	# of nets	Available buffers	EC	Grid	M	Average		# noise violation nets			Run time		
						MAN	MN	BS	G	B	BS	G	B
ami33	112	3011	9	30 × 33	6	3.2	1.5	0	31	46	5s	4s	8s
ami49	368	6889	16	30 × 33	7	4.2	1.5	0	66	103	12s	9s	20s
apte	77	1811	9	30 × 33	6	3.4	1.7	0	6	41	5s	3s	5s
hp	68	2386	9	30 × 33	5	3.5	2.2	0	14	21	3s	2s	4s
playout	1294	15884	56	30 × 33	7	18.0	5.5	0	165	568	51s	36s	69s
a9c3	1148	11847	44	30 × 33	6	13.0	5.6	0	106	481	37s	29s	50s
ac3	200	4034	14	30 × 33	6	4.6	2.4	0	20	85	9s	6s	12s
hc7	430	7938	26	30 × 33	7	7.8	4.0	0	59	122	13s	10s	22s
n30c	390	5513	15	30 × 33	7	4.3	1.5	0	58	149	11s	3s	13s
n50c	515	8404	17	30 × 33	6	4.2	1.8	0	55	140	10s	4s	15s
n100a	885	8931	32	30 × 33	6	7.2	4.3	0	68	348	25s	7s	25s
n200a	1585	13803	64	33 × 33	7	20.1	10.0	0	86	742	57s	14s	58s
n200b	1714	16903	65	33 × 33	6	19.6	9.4	0	278	850	63s	29s	48s
n300	1893	23295	68	33 × 33	7	19.9	8.4	0	159	879	70s	33s	54s
syn1	10086	87773	334	40 × 40	6	96.5	60.7	0	1057	4836	508s	331s	563s
syn2	10486	90577	348	40 × 40	6	102.8	61.3	0	1345	4963	637s	398s	558s

Table 3.2: Comparisons of routing and noise protection results. EC is the edge capacity; BS represents the simultaneous buffer and shield insertion algorithm; G represents the greedy algorithm; B represents the buffer-only algorithm.

We compare our results with those of two other methods. The first method is similar to the idea of [4], in which *only* buffers are inserted to reduce the noise, and the shielding effects are not considered. The buffers are also inserted by a dynamic programming-like algorithm, trying

to achieve the noise constraint with the fewest number of buffers. The second method that we compared against is a greedy approach, in which buffers and shields are assigned in separate stages. Buffers are first assigned in the same way as [5]. With the buffer positions known, we attempt to insert shield wires for each routing edge. For each net, the greedy shield insertion is composed of two steps:

1. We use a bottom-up approach, using every possible shield along the routing edges to meet the noise constraint.

2. If step 1 is successful, it may be the case that more than enough shields have been inserted. We then follow a top-down peel-off procedure to remove all of the unnecessary shields and buffers until the noise constraint or driving length constraint has been violated. The peel-off is greedy in the sense that a shield that is closer to the source of a tree will be removed greedily first. If there are multiple choices at any step in the top-down process, the branch with the higher noise margin will have its shield peeled off first.

Both of the above comparison methods employ the same routing and rerouting procedure as our algorithm.

The experimental results are listed in Table 3.2. The first eight columns show some basic properties of the circuits. Next, the results of our method which introduces buffers and shields (BS), the first comparison method which introduces buffers only (B), and the greedy buffering and shielding method (G) are shown. Empirically, results show that the asymptotic run time of our buffer and shield insertion algorithm is linear in the number of nets, and our algorithm scales easily to cases with over 10,000 nets. The B and G columns show that these methods can result in noise protection failures on as many as 53% (circuit *apte*) and 28% (circuit *ami*33) of the total number of nets. In comparison, our simultaneous shield and buffer insertion approach has achieved the protection goals successfully without much sacrifice in speed, and in all cases, all of the nets meet the noise constraints.

The buffer-only approach shows poor performance because of the restricted number of buffers that are available. In the competition for limited number of buffers, a large number of nets cannot

obtain enough buffer resources. Without the help of intelligent shield insertion, they fail the noise protection. As interconnects continue scaling, next generation technology will see a more intensified contention for buffers, and this becomes more of an issue for future interconnect design, as projected by [98]. The greedy approach, on the other hand, performs buffer insertion and shield protection in separate steps, and no concerns of noise constraint are considered in buffer insertion, resulting in an inferior performance to our integrated buffer and shield insertion solution.

Circuit	Overflow		Circuit	Overflow	
	BS	*G*		*BS*	*G*
ami33	0	416	*n30c*	0	271
ami49	0	583	*n50c*	0	398
apte	0	22	*n100a*	0	207
hp	0	168	*n200a*	0	168
playout	0	1774	*n200b*	0	4666
a9c3	0	981	*n300*	0	1604
ac3	0	228	*syn1*	0	30524
hc7	0	1054	*syn2*	0	36634

Table 3.3: Overflow of routing and protection if 100% protection is achieved.

An additional advantage of our approach is the adaptive supply net architecture, which enables a flexibility between the requirements of signal and power routing, so that both routing and supply net requirements are simultaneously met. For all three algorithms in Table 3.2, the routing overflow are almost 0 for all benchmarks, and is hence not listed. However, in cases where more nets must be protected to resolve the remaining noise violations, extra routing resources must be employed, leading to overflows. Table 3.3 reports the overflow results for our algorithm and the greedy algorithm if 100% protection is desired (the buffer-only algorithm does not use shield resources, and is omitted from this comparison). The results show that much more routing resources have to be sacrificed to obtain a good protection for the greedy algorithm, while our algorithm can successfully achieve a good protection without extra routing overflow.

3.6.2 Verification of Noise Margin Inflation

As described in Section 3.4.2, as an effective way to compensate for the pessimism of Devgan's metric, we can heuristically inflate the specified noise margin NM_{spec} to be $NM_{inf} \approx$

$a_0 NM_{spec} + a_1$ and take this inflated value as the input to our algorithm. It is less pessimistic this way, and thus fewer protection resources are required to accomplish protection. Furthermore, we want to show experimentally that this inflated noise margin is at an appropriate level, so that it does not cause false positive errors during the pruning process in our algorithm, i.e., a protection solution satisfying the inflated noise margin NM_{inf} under Devgan's metric should also satisfy the originally specified noise margin NM_{spec} when measured with SPICE simulation.

The coefficients for noise margin inflation are experimentally determined in Section 3.4.2 with least square fitting technique and $a_0 = 1.93$ and $a_1 = -0.26$V. With $NM_{spec} = 0.4$V in our experiment, we can obtain $NM_{inf} = 0.51$V. Inputting NM_{inf} to our algorithm, the generated protection solutions are then simulated with SPICE to verify whether they still satisfy the originally specified noise margin $NM_{spec} = 0.4$V. Due to long run time, we randomly selected up to 700 nets from each circuit for simulation; for smaller benchmarks such as *hp*, all nets were simulated. As a comparison, we further exaggerate the noise margin to $NM_{exag} = 0.6$V and input it to our algorithm. By performing the same SPICE verification process on the generated protection solutions, we compare the simulation results in Table 3.4.

Circuit	P_{inf}	P_{exag}	Circuit	P_{inf}	P_{exag}
ami33	100%	93.3%	*n30c*	100%	88.9%
ami49	100%	92.4%	*n50c*	100%	86.5%
apte	100%	89.5%	*n100a*	100%	92.8%
hp	100%	94.3%	*n200a*	100%	90.7%
playout	100%	92.4%	*n200b*	100%	94.2%
a9c3	100%	95.3%	*n300*	100%	94.4%
ac3	99.3%	94.6%	*syn1*	100%	94.2%
hc7	99.1%	93.1%	*syn2*	100%	95.7%

Table 3.4: Net protection rate with inflated NM_{spec}. P_{inf} and P_{exag} are the percentage of nets getting fully protected under SPICE simulation with inflated NM_{spec} at $NM_{inf} = 0.51$V and $NM_{exag} = 0.6$V, respectively.

In Table 3.4, P_{inf} is the percentage of nets that satisfy the inflated noise margin NM_{inf} under Devgan's metric, and also satisfies the specified noise margin NM_{spec}; P_{exag} is the percentage in the case for $NM_{exag} = 0.6$V. As can be seen, with NM_{spec} inflated to be $NM_{inf} = 0.51$V, almost 100% of the solutions can still satisfy the original $NM_{spec} = 0.4$V noise margin under accurate SPICE simulation; while this percentage drops to about 90% when NM_{spec} is further exaggerated

to $NM_{exag} = 0.6$V. This result suggests that with too much inflation of noise margin, we can obtain false protection solution which actually fails the original noise requirement in reality; while our proposed methodology for noise margin inflation in Section 3.4.2 is safe and can acquire a good yet economic solution.

3.7 Conclusion

We have shown in this chapter a method for simultaneously inserting supply shields and buffers during global routing to reduce crosstalk noise under a novel power supply architecture. The method employs a length-constraint based buffer insertion model that also naturally takes into account delay considerations, and uses a dynamic programming-like bottom-up method to optimally assign shields and buffers with the least resource usage. We have also shown the fidelity of Devgan's metric, and the noise inflation technique to effectively compensate for its pessimism. Experimental results show that this method can route nets to meet both capacity and noise constraints. It is more effective than noise reduction using a buffer-only approach or a greedy approach.

Chapter 4

Temperature-Aware Routing in 3D ICs

Three-dimensional integrated circuits (3D ICs) provide an attractive solution for improving interconnect performance and realizing higher degrees of integration by packing more transistors per unit volume and permitting heterogeneous integration of technologies in system-on-chip (SoC) design. The move to 3D is a topological change that is directly related to solving new physical design problems, but such solutions must be embedded in an electrothermally-conscious design methodology, since 3D chips generate a significant amount of heat per unit volume. In this chapter, we propose a temperature-aware 3D global routing algorithm with insertion of "thermal vias" and "thermal wires" to lower the effective thermal resistance of the material, thereby reducing the chip temperature. Since thermal vias and wires take up lateral routing space, our algorithm utilizes sensitivity analysis to judiciously allocate their usage, and iteratively resolve the contention between routing and thermal vias and wires. Experimental results show that our routing algorithm with thermal via and thermal wire insertion can effectively reduce the peak temperature to meet the temperature requirement as well as alleviate routing congestion.

4.1 Introduction

Three-dimensional integrated circuits (3D ICs) have attracted a great deal of attention in recent years. 3D integration attempts to overcome the problems associated with increased interconnect effects, most particularly those related to delay and power dissipation. By stacking multiple device layers into a monolithic structure, 3D ICs can achieve higher transistor densities and shorter interconnect lengths than two-dimensional (2D) ICs, and work as a platform to integrate different components such as digital ICs and analog ICs into one circuit stack, and provide good substrate isolation among them [11]. Despite the advantages of 3D ICs over 2D ICs, thermal effects are expected to be significantly exacerbated in 3D ICs due to higher power density and greater thermal resistance of the insulating dielectric, and this can cause greater degradation in device performance and chip reliability which have already plagued 2D ICs [11]. Thus, it is essential to develop 3D-specific design tools that take a thermal co-design approach so as to address the thermal effects and generate reliable and high performance designs. The work in this chapter considers the problem of developing routing solutions for 3D ICs.

To address thermal issues in global routing, an efficient and accurate thermal computation method is needed. Previous work on on-chip thermal analysis falls into the following categories. In [47, 78, 109], the finite difference method (FDM) and the finite element method (FEM) are used, respectively, to compute the on-chip temperature distribution. Each of these can handle different boundary conditions and non-uniform thermal properties in the medium. Another approach utilizes the Green function method as in [23, 121], which has the advantage of analyzing the temperature at points of interest with only local analysis, but lacks the ability to analyze a system with heterogeneous thermal conductivity. Our work employs thermal analysis to identify best locations for thermal via and thermal wire (which will be detailed described shortly) insertion, and will apply it in conjunction with adjoint sensitivity analysis in a thermally heterogeneous 3D IC. Hence, FDM, as a simple yet efficient method, is utilized in this chapter. A framework of using this method to perform thermal and sensitivity analysis has been presented in preliminary Sections 2.4.3 and 2.4.4.

3D global routing algorithms have been studied in [2, 33, 37, 39, 44]. In [44], a 3D channel routing algorithm is proposed, while [2] and [39] perform 3D maze routing with congestion control for 3D ICs and 3D Field-Programmable Gate Arrays (FPGAs), respectively. In [37], a hierarchical routing algorithm is applied to 3D circuit and treats a 3D IC as a set of aligned 2D routing regions. However, none of these approaches considers the thermal problems associated with 3D ICs in the routing phase. A first approach to this problem is presented as a thermally-driven 3D routing algorithm in [33] with the planning of thermal vias to reduce temperature. However, it does not fully address the contention issues between thermal vias and routing resources.

In this chapter, we propose a novel 3D routing algorithm which can effectively reduce on-chip temperatures by appropriate insertion of "thermal vias" and a new construct that we call "thermal wires," and generate a routing solution free of thermal and routing capacity violations. *Thermal vias* correspond to vertical interlayer vias that do not have any electrical function, but are explicitly added as thermal conduits. *Thermal wires* perform a similar function, but conduct heat laterally within the same layer. Thermal vias perform the bulk of the conduction to the heat sink, while thermal wires help distribute the heat paths over multiple thermal vias. The routing scheme begins with routing congestion estimation and signal interlayer via assignment, followed by thermally-driven maze routing. Sensitivity analysis is employed and linear programming (LP) based thermal via/wire insertion is performed to reduce temperature. The above process is iteratively repeated until temperature and routing capacity violations are resolved. Experimental results show that the scheme can effectively resolve the contentions between thermal via/wire and routing, generating a solution satisfying both congestion and temperature requirements.

The rest of the chapter is organized as follows. Section 4.2 details the interconnect model and in 3D ICs. The temperature-aware 3D global routing problem is formulated in Section 4.3. Section 4.4 describes the thermal circuit model used in our routing algorithm. Section 4.5 presents the 3D global routing algorithm with sensitivity analysis and thermal via/wire insertion. This is followed by experimental results in Section 4.6 and conclusion in Section 4.7.

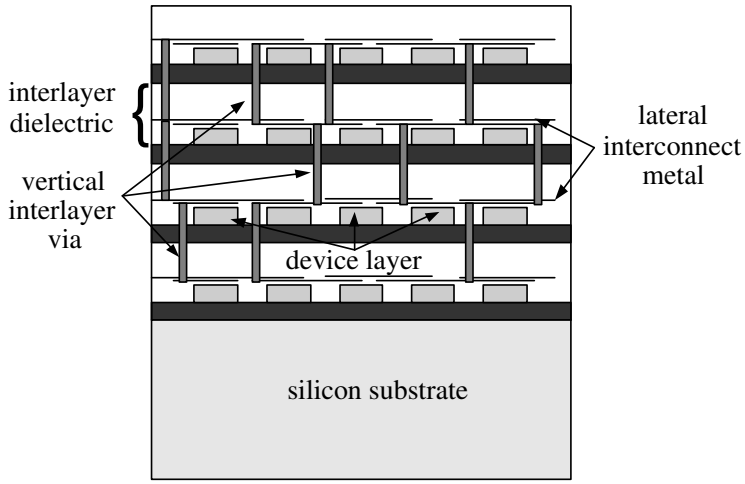


Figure 4.1: Cross section of a 3D circuit with four active device layers.

4.2 Interconnects in 3D ICs

4.2.1 3D IC Routing Model

In 3D ICs, devices are fabricated on a number of active layers. Figure 4.1 shows the cross section of a 3D circuit with four active device layers, which are separated by silicon dioxide and joined by an adhesive material. Within each device layer, interconnections among devices can be achieved with traditional interconnect wires and vias. Connections between active layers are facilitated by vertical interconnect vias that span through multiple layers, providing a means for electrically connecting wires in those layers. This type of via is different from a regular 2D via: in particular, it is significantly taller than a conventional via, and has a larger landing pad to maintain a viable aspect ratio. Such vias are shown as shaded vertical pillars in Figure 4.1 and we refer to them as *interlayer vias* after Section 2.2.2.

The multiple-layer structure of 3D IC complicates the global routing problem, but we may extend some basic constructs from 2D routing to 3D. The 3D global routing grid and global routing graph can be built as in Section 2.2.2. Essentially the 3D global routing graph is an stacking of the 2D global routing graphs for all active layers, with vertical edges connecting neighboring ones, which represents interlayer vias. A net N in a 3D IC consists of a set of electrically equivalent pins, $\{n_0, n_1, n_2, \ldots, n_k\}$, distributed in different GRCs (possibly on different active layers), of which

n_0 is the source and $n_1, n_2, ..., n_k$ are sinks. If a net has pins distributed on different layers, it is referred to as an *3D net*; otherwise it is called a *2D net*. The global routing of a net in a 3D IC is to find a set of edges that connect all pins in the corresponding 3D global routing graph.

4.2.2 Thermal Vias and Thermal Wires

The silicon dioxide layer acts as a thermal insulator that strongly inhibits heat flow, potentially leading to elevated temperatures. Interlayer vias connecting different active layers serve to to conduct heat and alleviate hot spots in 3D ICs [24, 25]. Straightforward extensions of traditional routing scheme to 3D are inadequate, since they leave out several new 3D-specific complications. The locations of the interlayer vias that carry signals from one layer to another should not be determined purely by the signal wire routing process, but should also incorporate thermal considerations, determining the most desirable interlayer via positions from the point of view of heat conduction. Moreover, a sufficient number of interlayer vias must be used so that the temperature can be reduced to below the target level. Our work uses two types of vias:

- *thermal vias* are deliberately introduced interlayer vias that serve no electrical function, but are dedicated to the purpose of temperature reduction [48], and
- *signal interlayer vias*, which carry electrical signal, and thus perform signal and heat conduction simultaneously.

Like signal interlayer vias, thermal vias can be planned in each GRC position i between two layers. Following the vertical edge capacity constraint for 3D global routing graph defined in Section 2.2.2, we require $V_i \leq U_i$, in which U_i is the interlayer via capacity for GRC i, and V_i is the total number of *both* signal interlayer vias *and* thermal vias going through GRC i. Here we assume that the interlayer vias for power grid are predetermined and there are dedicated resources for them, which are excluded from U_i.

Although interlayer vias can effectively reduce the on-chip temperature, their large size also acts as a significant routing blockage; as more heat removal is achieved, less routing space is available, and this problem has been identified in the packaging of multi-chip modules (MCMs) [75].

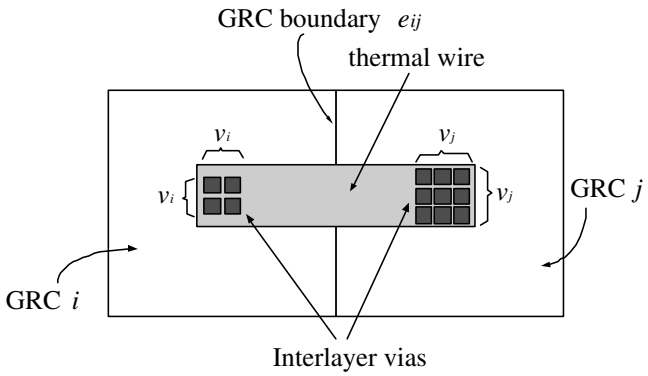


Figure 4.2: Reduction of lateral routing capacity due to the interlayer vias in neighboring GRC; thermal wires are lumped, and together with thermal vias, form a thermal dissipation network.

Figure 4.2 shows how interlayer vias can reduce the routing capacity of a neighboring lateral routing edge. If $v_i \times v_i$ interlayer vias pass through GRC i, and $v_j \times v_j$ interlayer vias pass through the adjacent GRC j, the signal routing capacity of boundary e_{ij} will be reduced from the original capacity, C_e, to a reduced capacity, $C_{e,red.}$, and we require that signal wire usage W_e should satisfy:

$$C_{e,red.} = \min\ (C_e - v_i \cdot w, C_e - v_j \cdot w)$$

$$W_e \leq C_{e,red.} \tag{4.1}$$

where w is the geometrical width of an interlayer via. Here we define the smaller of the two reduced routing widths as the reduced edge capacity, so that there can be a feasible translation from the global routing result to a detailed routing solution. On the other hand, given the actual signal wire usage W_e of a routing edge, we can also use equation (4.1) to determine how many interlayer vias can go through the neighboring GRCs so that there is no overflow at the routing edge. Since temperature reduction requires insertion of a large number of thermal vias, careful planning is necessary to meet both the temperature and routability requirements.

Just as thermal vias enhance vertical heat conduction, we can also introduce *thermal wires* to improve lateral heat conduction. The lateral routing tracks in the circuits are used by power grid wires and signal wires. For simplicity of analysis, we assume a predetermined power grid

architecture as well as dedicated routing tracks for power wires, and they are excluded from the routing capacity calculation. Along a routing edge, signal wires may not utilize all of the routing tracks, and we employ the remaining tracks to connect the thermal vias in laterally adjoining GRCs with *thermal wires*. Like thermal vias, thermal wires do not carry signal, and therefore, they can be connected directly to thermal vias to form an efficient heat dissipation network as shown in Figure 4.2. Thermal wires enable the conduction of heat in lateral direction, and can thus help vertical thermal vias to reduce hot spots temperature efficiently: for those hot spots where only a restricted number of thermal vias can be added, we can use thermal wires to conduct heat laterally, and then remove heat through thermal vias in adjoining grids. Note that although signal wires can also conduct heat laterally, they must be separated from the thermal vias by an oxide that has high thermal resistivity, and are therefore not efficient in lateral heat removal. Moreover, unlike signal routes that only use a small number of interlayer signal vias, the thermal vias form a global net and are therefore very effective in heat removal. Another advantage of thermal wires is improved design for manufacturing. Filling the remaining tracks with thermal wires can create a dense metal density distribution for the chemical-mechanical polishing (CMP) step, which improves manufacturability. Since thermal wires contend for lateral routing resources with signal wires, they should be well planned to satisfy the temperature and routability requirements. Along a routing edge e with total routing capacity C_e, the number of signal wires s_e and the number of thermal wires m_e should satisfy

$$(s_e + m_e) \cdot w_t \leq C_e \tag{4.2}$$

where w_t is the width of a routing track.

4.3 Temperature-Aware 3D Global Routing Problem

Our formulation of the temperature-aware 3D global routing problem is as follows. The inputs to the problem include:

1. The placement of a 3D IC

2. 3D technology parameters

3. A specified value of the maximum temperature, T_{spec}

Our global routing algorithm will route the circuit netlist and efficiently insert thermal vias and thermal wires at appropriate locations so that the solution satisfies the requirements on the routing capacities and the interlayer via capacities. At the same time, the peak temperature will be lowered towards T_{spec}. Moreover, for reasons related to reliability and to the limited availability of interlayer via resources, we require that the number of signal interlayer vias used in routing should be minimized.

4.4 Thermal Analysis in 3D ICs

4.4.1 Thermal Analysis Model

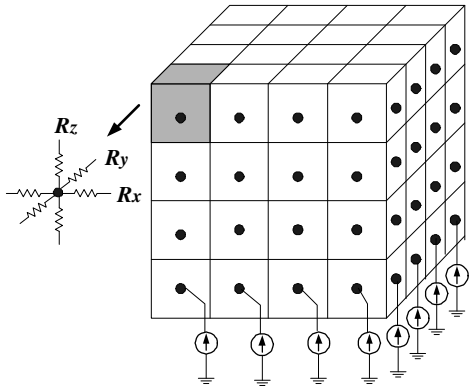


Figure 4.3: Equivalent thermal circuit model of a 3D IC.

Equivalent thermal circuit models have been used extensively in the simulation of thermal effects because of their clear physical origins, ease of implementation, and flexibility in handling different kinds of boundary conditions and non-uniform thermal conductivity. Our thermal analysis employs the finite difference method (FDM) to build thermal circuit following the general thermal modeling framework presented in Section 2.4.3. To facilitate thermal analysis for 3D routing purpose, our thermal model uses a grid of the same size as the 3D global routing grid. We show

the tessellation and thermal modeling for a 3D IC in Figure 4.3. As described in Section 2.4.3, we use a node to represent all of the heat dissipating devices in a GRC, and employ an equivalent current source to represent the total power generated in the GRC. Adjacent nodes are connected by a thermal resistance, forming a thermal resistive grid. In this chapter, we only consider power dissipation from active devices, since heat generated by interconnect wires is small because of their relatively small resistance; however, it is possible to extend the approach to incorporate this. This way, a 3D chip is modeled as a resistive thermal network and we can solve the corresponding linear equation to obtain the temperature of each node.

The numerical value of thermal resistance connecting to a node in x, y, and z directions can be computed as:

$$\begin{aligned} R_x &= R_{oxide,x}||R_{int.,x}||R_{mix,x} \\ R_y &= R_{oxide,y}||R_{int.,y}||R_{mix,y} \\ R_z &= R_{oxide,z}||R_{int.\ via,z}||R_{mix,z} \end{aligned} \tag{4.3}$$

where the symbol "||" refers to a parallel connection between the resistors. The thermal resistance in each direction is determined by the materials contained in the cuboid volume extending to the neighboring GRC and can be calculated as parallel connections of three parts. $R_{oxide,x}$, $R_{oxide,y}$ and $R_{oxide,z}$ are the oxide thermal resistance of pure oxide extending to neighboring node in x, y and z directions respectively. The terms $R_{int.,\ x}$ and $R_{int.,\ y}$ are the thermal resistance of interconnect wires running along x and y directions, respectively. $R_{int.\ via,\ z}$ is the thermal resistance of the vertical interlayer vias which act as one of the major heat conduction paths, and it is proportional to the number of interlayer vias connecting the vertically neighboring GRCs. Along each direction, besides pure oxide and metal wires or vias extending in that direction, the remaining volume is a mixture of metal wires/vias extending in the orthogonal directions and the oxide separating them; along the direction we are considering, alternate metal and oxide form a "sandwich" structure. We denote the expected thermal resistance of such a mixture as $R_{mix,x}$, $R_{mix,y}$ and $R_{mix,z}$ for x,

y and z direction respectively, and their values are calculated as series thermal resistance of the metal wires/vias (running in orthogonal directions) and the oxide which separates the metal.

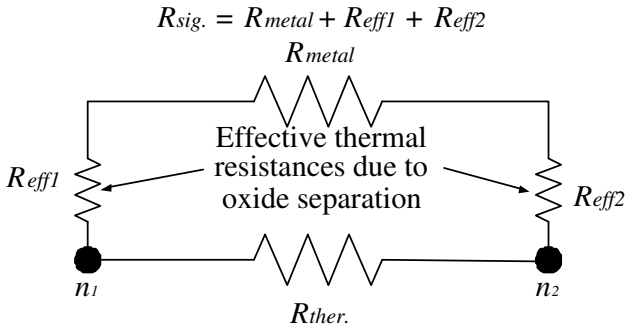


Figure 4.4: Thermal circuit model of signal wire and thermal wire.

To differentiate between the the heat conducting abilities of signal wires and thermal wires, we use the model depicted in Figure 4.4. We model a signal wire of thermal resistance $R_{sig.}$, connecting neighboring GRCs n_1 and n_2 as series connection of metal resistance R_{metal} and two additional resistances R_{eff1} and R_{eff2} connecting to the thermal via nodes at the center of n_1 and n_2. R_{eff1} and R_{eff2} correspond to the effective thermal resistance of the oxide which separates signal wires from thermal vias. The effective oxide thickness is estimated by a probabilistic approach which assumes a uniform distribution of signal wires crossing a GRC boundary. By comparison, the thermal resistance of a thermal wire connecting n_1 and n_2 only consists of the metal resistance $R_{ther.}$.

The vertical edges of the 3D chip are modeled to be adiabatic to the ambient, and the bottom of the chip is connected to an isothermal heat sink by a bulk substrate. With the thermal circuit established, like Section 2.4.3, nodal analysis (NA) equations for the thermal grid can be set up as

$$\mathbf{MT} = \mathbf{P} \tag{4.4}$$

where $\mathbf{M}$, $\mathbf{T}$ and $\mathbf{P}$ are the NA coefficient matrix, the temperature profile vector and power dissipation vector for the GRCs, respectively. This equation is sparse and symmetric positive definite

(SPD), and we use the LASPACK package [104] as the linear solver to obtain the temperature profile efficiently.

4.4.2 Thermal Vias and Thermal Wires Placement

To effectively remove temperature violations in circuit under constrained resources, it is very important that we insert thermal vias and thermal wires in the *right places*; otherwise even a large number of them can not effectively reduce the high temperatures.

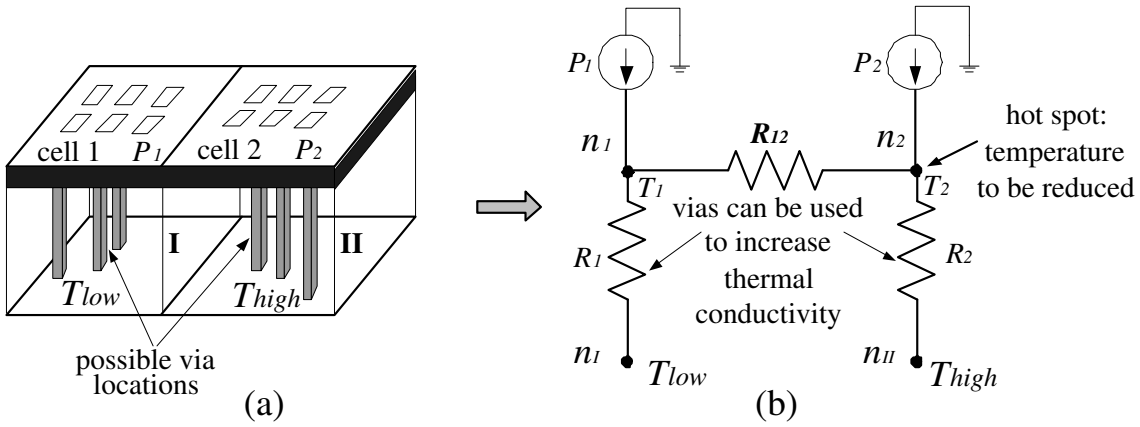


Figure 4.5: A toy example of interlayer via insertion for hot spot temperature reduction.

To illustrate this point, we take thermal via insertion as an example and shows a simple scenario in Figure 4.5. Figure 4.5(a) shows two grid cells, cell 1 and cell 2, neighboring each other, and the active devices within them generate a power of P_1 and P_2, respectively, with $P_1 \ll P_2$. Geometrically, these two cells are vertically above region I, with low temperature T_{low}, and region II, with high temperature T_{high}. The temperature at cell 1 and cell 2 are denoted as T_1 and T_2, respectively. As there is much more power generated in cell 2, we aim at reducing the hot spot temperature T_2 by inserting interlayer vias in region I or II, and evaluating the effectiveness of different interlayer via insertion schemes. The equivalent thermal circuit is built following Section 2.4.3 and is shown in Figure 4.5(b). It consists of four nodes n_1, n_2, n_I and n_{II} representing cell 1, cell 2, region I and region II, respectively. Using the method described in Section 4.4.1, we can extract the thermal resistances R_{12}, R_1 and R_2 shown in Figure 4.5(b). R_{12} is the thermal resistance between cell 1 and cell 2, and R_1 and R_2 are the thermal resistance between

cell 1 and region I, as well as the thermal resistance between cell 2 and region II, respectively.

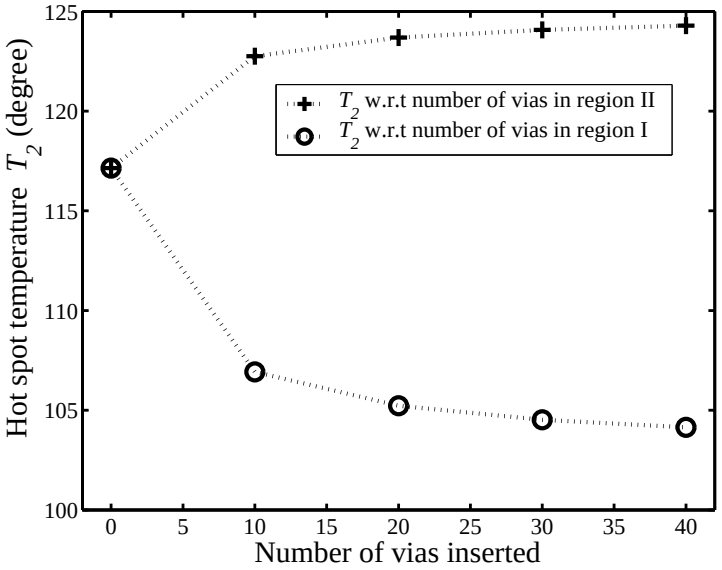


Figure 4.6: Changes of hot spot temperature (T_2) due to interlayer via insertion in region I and II of Figure 4.5.

The insertion of thermal vias in region I and region II can significantly reduce the value of R_1 and R_2, hence affect the hot spot temperature T_2. Nevertheless, it is the location at which thermal vias are inserted that plays a very important role in reducing the temperature at interest. In our example, under the scenario $T_{high} >> T_{low}$, it is far more effective to insert thermal vias in region I than region II. Let the power density for cells be $P_1 = 20\text{W/cm}^2$ and $P_2 = 400\text{W/cm}^2$, and T_{high} =125˚C, T_{low} =20˚C. Temperature analysis is performed under the same experimental setup as in Section 4.6 and we show the relationship between the hot spot temperature T_2 and the number of vias inserted in different regions as curves in Figure 4.6. The curves show that insertion of thermal vias in region I can efficiently reduce the hot spot temperature T_2 (as the lower curve shows), while a large number of via insertions in region II (although right below hot spot) do not reduce the hot spot temperature T_2 at all (as the higher curve shows). Even worse, the insertion of more thermal vias in region II increases temperature T_2! The intuition behind this phenomenon is that region II has a very high temperature, and build a better thermal conduction path (by insertion of more thermal vias) to this region actually help heat to flow *to* n_2 and thus raise temperature T_2, instead of lowering the temperature as we desired. On the other hand, thermal

via insertion in region I can help improve the thermal conduction along the path $n_2 \rightarrow n_1 \rightarrow n_I$, and since node n_I has a low temperature T_{low}, this thermal via insertion can efficiently help heat dissipation of cell 2.

This simple example shows the importance of inserting thermal vias at the *right places*, so that they can best exert their power in heat dissipation, and it also applies to the thermal wire insertion scenario. In our routing, a big concern is the constraints of resource availability and contention between thermal vias, thermal wires, and lateral routing, therefore insertion of thermal vias and wires at the *right places* becomes especially critical to meet both thermal and routability requirements. Hence it motivates a quantitative sensitivity analysis to identify the best locations for thermal via/wire insertion.

4.4.3 Adjoint Sensitivity Analysis for 3D ICs

Thermal vias and thermal wires play an important role in heat conduction, but their distribution is constrained by white space and lateral routing usage. Moreover, as discussed in last section, it is important to place them at the *right* places. Thus, we perform adjoint sensitivity analysis following the framework described in Section 2.4.4, and this can help determine the potential thermal via and thermal wire locations where insertion of a thermal via/wire can reduce hot spot temperature effectively. Adjoint sensitivity analysis has the advantage of obtaining the sensitivities of one output value to many parameters simultaneously and efficiently. Applying the adjoint sensitivity analysis method in Section 2.4.4 to our 3D thermal problem, we calculate the the sensitivity of the temperature at a hot spot to the thermal via density as an example; sensitivity analysis for hot spot temperature to thermal wire density can be performed similarly.

The temperature profile of a 3D circuit is determined by equation $\mathbf{MT} = \mathbf{P}$ as in Section 4.4.1, and we are interested in a hot GRC in the circuit with temperature T_i at the i^{th} position of vector $\mathbf{T}$. Applying the derivations in Section 2.4.4 to our 3D IC thermal problem, we can find that the temperature vector $\mathbf{T}$ and power vector $\mathbf{P}$ respectively correspond to the vectors $\mathbf{x}$ and $\mathbf{b}$ in equations (2.7), (2.8) and (2.9), and the scalar performance function $f(\mathrm{x}) = T_i$.

Plugging these into equation (2.9), we can obtain the sensitivity of T_i to matrix element M_{kl} is expressed as follows:

$$\frac{\partial T_i}{\partial M_{kl}} = -\xi_{ik} T_l \tag{4.5}$$

in which $\xi_{\mathbf{i}}$ is the solution of equation $\mathbf{M}^T \xi_{\mathbf{i}} = \mathbf{e_i}$, and $\mathbf{e_i}$ is a column vector with all zeros except for a one in its i^{th} row. We can then obtain the sensitivity s_{ij} of the temperature T_i to the number of interlayer vias at location j, V_j, by applying the chain rule:

$$s_{ij} = \frac{\partial T_i}{\partial V_j} = \sum_{M_{kl} \in S_j} \frac{\partial T_i}{\partial M_{kl}} \frac{\partial M_{kl}}{\partial V_j} \tag{4.6}$$

where set S_j contains all of the entries, M_{kl}, that depend on V_j. There is a small number of such entries in each S_j, since the V_j term only appears in the vertical and lateral thermal conductance related to this location (thus, there are at most 12 entries in each S_j since each conductance appears four times in matrix $\mathbf{M}$). This sensitivity calculation indicates how effective thermal via insertion can be in reducing high temperatures, and we can use it to guide thermal via insertion. Similarly, we can perform sensitivity analysis for hot spot temperature to thermal wire density and identify the most effective location for thermal wire insertion.

4.5 3D Temperature-Aware Global Routing

In addition to the traditional 2D-like challenges of wire length and congestion, 3D global routing must deal with the complexity introduced by higher dimensional routing and by temperature constraints. Our global routing approach resolves these challenges in two phases. Phase I first performs signal interlayer via assignment, and then utilizes a thermally-driven 2D maze routing algorithm in each layer to generate an initial routing solution. Based on these results, phase II resolves temperature and congestion violations iteratively: it identifies hot spots and sensitivity information, and then judiciously inserts thermal vias and thermal wires in sensitive locations;

the insertions, together with rip-up-and-reroute are iteratively performed until both congestion and temperature violations are resolved. In the above rip-up-and-reroute, we process one net at a time and maintain a fixed order of all nets. We have experimentally found that, under different randomly chosen net orderings, the results change very little as long as we maintain the same fixed net order through all of the iterations. This is due to the fact that early iterations are seen to create good estimates of resource utilization, and this reduces the order dependence. This is consistent with the observation in [86]. Since thermal effects pose a significant problem in 3D ICs and the thermal via and thermal wire insertion for temperature reduction is a core part of our algorithm, we will first describe routing phase II in this chapter; the initial routing procedures of phase I and the overall flow are discussed following that.

4.5.1 Thermal Via and Thermal Wire Insertion for Temperature Reduction

In an initial global routing solution generated from phase I, the high power density and low thermal conductivity of silicon dioxide can result in hot spots in 3D ICs, and we can employ thermal vias and thermal wires to build effective heat conduction paths. Assuming a predetermined distribution of wires and interlayer vias for power lines, we can analyze the heat conduction for a 3D IC. The role of the thermal via and thermal wire insertion algorithm is to reduce the peak temperature towards T_{spec}, and at the same time, generate a routable design. This task must take into account the routing blockages and resource utilization (via capacity and wire capacity) as these are added. Our algorithm iteratively inserts thermal vias and thermal wires with a linear programming (LP) approach, and performs a rip-up-and-reroute step to obtain a solution free from temperature and congestion violations.

Thermal Via and Thermal Wire Insertion Algorithm

Figure 4.7 shows the flow of the algorithm. The LP-based thermal via and thermal wire insertion and rip-up-and-reroute are iteratively performed until there is no violation or no further

Algorithm: `Thermal_via/wire_insertion_for_temperature_reduction`

Input: Initial 3D global routing result and specified peak temperature requirement T_{spec}.
Output: A routing solution with thermal via and thermal wire insertion that is free from congestion and temperature violation.

```
begin
  do
  {
     Update temperature profile, get current highest temperature T_max;
     Update target temperature T_target according to T_max as described in Section 4.5.1;
     Perform sensitivity analysis for temperature violating spots with T > T_target;
     Construct and solve LP formulation: equations (4.7), (4.8), (4.9), (4.10) and (4.13);
     Insert thermal vias and thermal wires and update congestion information;
     Rip-up-and-reroute for resolving overflow;
  } until (free from temperature and congestion violation or no further improvement)

  Post insertion of thermal vias and thermal wires using all available space;
end
```

Figure 4.7: Thermal via and thermal wire insertion algorithm for temperature and congestion reduction.

improvement. The latter is easily identified if it is detected that the congestion map changes insignificantly, or the peak temperature reduces trivially. In each iteration, we relax the specified temperature T_{spec} to a target temperature $T_{target} = T_{max}^{\mu} \cdot T_{spec}^{1-\mu}$ in identifying temperature violation, where T_{max} is the current highest temperature, and μ is a constant with value $0 < \mu < 1$ (we take $\mu = 0.2$ in practice); also if this calculated value of T_{target} is smaller than $T_{spec} + \Delta T_{min}$, we assign $T_{target} = T_{spec}$, where ΔT_{min} is a positive constant. By introducing ΔT_{min} we can avoid infinite number of temperature decreases before T_{target} reaches T_{spec}; in practice, we set $\Delta T_{min} =$ 5°C. As the final step we greedily insert thermal vias and thermal wires using all of the remaining space. Experimental results show that our algorithm can economically make use of thermal vias/wires and routing space, and generate an optimized routing solution.

Linear Programming Based Thermal Via and Thermal Wire Insertion

In the thermal via and thermal wire insertion algorithm of Figure 4.7, we first perform thermal analysis to update the temperature profile and identify hot spots that violate the temperature specifications. The thermal via and thermal wire planning procedure is then formulated as a LP problem to reduce the temperature of hot spots and minimize total thermal via and thermal wire usage, while leaving ample space for lateral routing.

For each of n temperature-violating hot spots with temperature $T_i > T_{target}$, $i = 1, 2, ..., n$,

we perform sensitivity analysis for T_i with respect to the number of thermal vias at each thermal via location as in Section 4.4, and this can be performed fast due to the advantage of adjoint sensitivity analysis. However, to reduce the complexity of the LP problem, we only record those non-trivial sensitivity values $s_{v,ij} \geq s_{th}$, in which s_{th} is a constant threshold of sensitivity and in practice we take $s_{th} = 0.01^{\circ}$C per via; the associated location j is then defined as a candidate thermal via location, and $j = 1, 2..., p$, where p is the total number of candidate thermal via locations. Similarly, we can define candidate thermal wire location and obtain the sensitivities $s_{w,ik}$ for T_i with respect to the number of thermal wires at each candidate location k, and $k = 1, 2, ..., q$, in which q is the total number of such locations. Based on the sensitivity analysis results, we can economically plan the number of inserted thermal vias, $N_{v,j}$, at each candidate thermal via location j, and the number of inserted thermal wires, $N_{w,k}$, at each candidate thermal wire location k, so that the temperature at each hot spot i can be reduced by ΔT_i. The LP problem is formulated as follows:

$$\text{minimize} \quad \sum_{j=1}^{p} N_{v,j} + \sum_{k=1}^{q} N_{w,k} + \Gamma \sum_{i=1}^{n} \delta_i \tag{4.7}$$

$$\text{subject to:} \sum_{j=1}^{p} -s_{v,ij} N_{v,j} + \sum_{k=1}^{q} -s_{w,ik} N_{w,k} + \delta_i \geq \Delta T_i,$$

$$i = 1, 2, ..., n, \ \Delta T_i = T_i - T_{target} \tag{4.8}$$

$$N_{v,j} \leq \min((1+\beta) R_{v,j}, \ U_j - V_j), \ j = 1, 2, ..., p \tag{4.9}$$

$$N_{w,k} \leq (1+\beta) R_{w,k}, \ k = 1, 2, ..., q \tag{4.10}$$

$$\delta_i \geq 0, \ i = 1, 2, ..., n; \tag{4.11}$$

$$N_{v,j} \geq 0, \ j = 1, 2, ..., p; \tag{4.12}$$

$$N_{w,k} \geq 0, \ k = 1, 2, ..., q \tag{4.13}$$

The objective function that minimizes the total usage of thermal vias and wires is consistent with the goal of routing congestion reduction. To guarantee the problem is feasible, we introduce relaxation variables δ_i, $i = 1, 2...n$. Γ is a constant which remains the same over all iterations. It

is chosen to be a value that is large enough to suppress the value of δ_i to be 0 when the thermal via and thermal wire resources in constraint (4.9) and (4.10) are enough to reduce temperature as desired.

Constraint (4.8) requires the temperature reduction at hot spot i, plus a relaxation variable δ_i, to be at least ΔT_i during the current iteration, where ΔT_i is the difference between current temperature T_i and target temperature T_{target}. Constraints (4.9) and (4.10) are related to capacity constraints on the thermal vias and thermal wires, respectively, based on lateral boundary capacity overflows in the same layer, and interlayer via capacity overflows across layers. Constraint (4.9) sets the upper limit for the number of thermal via insertions $N_{v,j}$ with two limiting factors. $R_{v,j}$ is the maximum number of additional thermal vias that can be inserted at location j without incurring lateral routing overflow on neighboring edge, and it is calculated as $R_{v,j} = v_j - v_{cur.,j}$, in which $v_{cur.,j}$ is the current interlayer via usage at location j, and v_j is the maximum number of interlayer vias that can be inserted at location j without incurring lateral overflow which can be calculated from equation (4.1). Adding more interlayer vias in the most sensitive locations can be very influential in temperature reduction, and therefore, we intentionally amplify the constraint by a factor β to temporarily permit a violation of this constraint, but which will allow better temperature reduction. This can potentially result in lateral routing overflow after the thermal via assignment, but this overflow can be resolved in the iterative rip-up-and-rerouting phase; in practice, we find $\beta = 0.1 \sim 0.2$ works well. A second limiting factor for $N_{v,j}$ is that the total interlayer via usage can not exceed U_j, which is the interlayer via capacity at position j, and we take the minimum of the two limiting factors in the constraint formulation. Similarly, constraint (4.10) sets a limit on the number of thermal wire insertions with the consideration of lateral routing overflow. $R_{w,k}$ is the maximum number of additional thermal wires that can be inserted at location k without incurring lateral routing overflow, and it is calculated as $R_{w,k} = m_k - m_{cur.,k}$, where $m_{cur.,k}$ is the current thermal wire usage at location k, and m_k is the maximum number of thermal wires at location k without incurring lateral overflow, which can be calculated from equation (4.2). In the same spirit of encouraging temperature reduction, we relax $R_{w,k}$ by factor of β, and any

potential overflow will be resolved in the rip-up-and-rerouting phase.

The LP problem above can be efficiently solved with the LP_SOLVE package [15]. The solution to this LP formulation are rounded to the ceiling integer to guarantee the optimality, and the corresponding number of thermal vias and thermal wires are then inserted into the 3D circuit; the resulting overflow can be resolved in the following rip-up-and-reroute. This LP based insertion and rip-up-and-reroute are iteratively performed until a feasible solution is reached.

4.5.2 Temperature-Aware 3D Global Routing Flow

We now describe the steps involved in our 3D routing algorithm. Briefly, these include a Minimum Spanning Tree (MST) generation and routing congestion estimation step, a signal interlayer via assignment step, and a thermally-driven 2D maze routing in each active layer; these steps constitute phase I of our routing algorithm. The phase II step is iterative routing involving rip-up-and-reroute and LP-based thermal via/wire insertion which has been described above in Section 4.5.1.

3D MST and Routing Congestion Estimation

A 3D net $N = \{n_0, n_1, n_2, ..., n_k\}$ can have more than two pins distributed on different layers (if not, as described earlier, we refer to it as a 2D net). We first build a minimum spanning tree, using Prim's algorithm [35], for each multiple-pin net, so as to decompose it into a set of two-pin nets. The cost function for two pins n_i and n_j of a net N in Prim's algorithm is defined as $C_{ij} = |x_i - x_j| + |y_i - y_j| + \alpha|z_i - z_j|$, where α is a large weighting factor that is used to discourage the use of two-pin interlayer nets, which can result in interlayer vias. In this expression, (x_i, y_i) and (x_j, y_j) are the 2D coordinates of n_i and n_j in terms of grid cell length; z_i and z_j are the layer numbers of the two pins. Therefore, the usage of this cost function can reduce total wire length and minimize the use of signal interlayer vias.

With a set of two-pin nets from MST decomposition, we can statistically estimate the routing congestion over each lateral routing edge using the L-Z shape statistical routing model proposed

in [113]. To extend the model in [113] for 3D routing, we assume that a two-pin net with pins on different layers has an equal probability of utilizing any interlayer via position within the bounding box defined by the pins. The estimated congestion map provides important information for the later signal interlayer via assignment stage, which simultaneously assign signal interlayer vias for all the interlayer nets.

Signal Interlayer Via Assignment

The 3D MST step can generate a number of two-pin interlayer decompositions, and our 3D routing algorithm must adjudicate their contention for the signal interlayer via locations, so that the later routing phase can generate high quality solution based on these assignments. To eliminate net order dependence, we simultaneously assign the signal interlayer vias for all interlayer nets under the the interlayer via capacity constraint, minimizing a cost function that reflects the wirelength and congestion along the routing path. However, some interlayer nets may go through more than two layers, and may require two or more interlayer vias between different layers to be assigned. Under such circumstances, the problem turns out to be a multi-layer assignment problem, which is proved to be NP-hard [90].

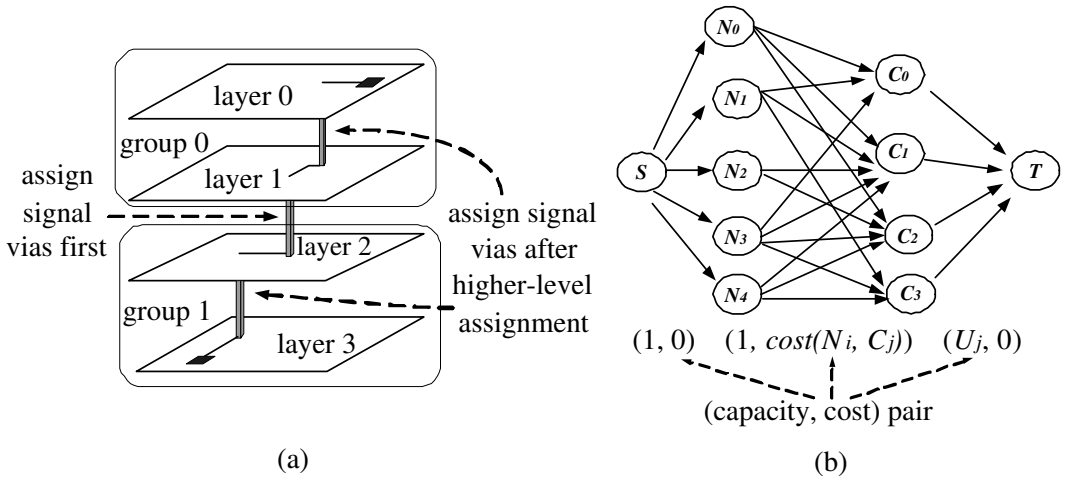


Figure 4.8: (a) Example of hierarchical signal via assignment for a four-layer 3D circuit. (b) Example of min-cost network flow heuristics to solve signal via assignment problem at each level of hierarchy.

Based on the two-pin interlayer decompositions generated from 3D MST step, our approach

uses a hierarchical min-cost network flow heuristic to simultaneously assign interlayer vias for all interlayer nets and minimize a cost function. First, this heuristic builds a group hierarchy by recursively dividing all device layers into two neighboring groups of equal size. Signal interlayer via assignment is then performed at the boundaries of group pairs at each level in a top-down way following the hierarchy: the assignment is conducted for the topmost level group boundary first, and then at the boundaries of group pairs of lower levels in the hierarchy. Figure 4.8(a) shows an example of signal interlayer via assignment for a decomposed 2-pin signal net in a four-layer circuit with two levels of hierarchy. The signal interlayer via assignment is first performed at the boundary of group 0 and group 1 at topmost level, and then it is processed for layer boundary within each group.

At each level of the hierarchy, the problem of signal interlayer via assignment is formulated as a min-cost network flow. Figure 4.8(b) shows the network flow graph for assigning signal interlayer vias of five interlayer nets to four possible interlayer via positions. Each interlayer net is represented by a node N_i in the network flow graph; each possible interlayer via position is indicated by a node C_j. If C_j is within the bounding box of the two-pin interlayer net N_i, we build a directed edge from N_i to C_j, and set the capacity to be 1, the cost of the edge to be $cost(N_i, C_j)$. The $cost(N_i, C_j)$ is evaluated as the shortest path cost for assigning interlayer via position C_j to net N_i when both pins of N_i are on the two neighboring layers; otherwise it is evaluated as the average shortest path cost over all possible unassigned signal interlayer via positions in lower levels of the hierarchy. The shortest path cost is obtained with Dijkstra's algorithm [35] in the 2D congestion map generated from the previous estimation step, and the cost function for crossing a lateral routing edge is a combination of edge length and a overflow:

$$cost = \text{ edge length} + \lambda \times \text{ overflow cost} \tag{4.14}$$

cost function which is similar to that in [51]: it assigns unit cost to an edge until edge usage reaches 80% of edge capacity, and increases cost linearly (to a maximum cost of 10) until edge usage reaches 40% above edge capacity. The incorporation of overflow into cost function can guide

signal interlayer via assignment to find the best location for minimizing routing congestion in each layer. The network flow graph in Figure 4.8(b) also includes a source node S and a target node T. There is an edge from S to every N_i with capacity 1 and cost 0; an edge is built from every C_j to target T with capacity U_j (the interlayer via capacity at C_j) and cost 0. There is a supply of N for source node S and a demand of N for target node T, where N is the number of interlayer net nodes in the flow graph. For the network flow graph, the min-cost network flow problem is optimally solved in polynomial time with the *cs2* package [46], and we can thus obtain the signal interlayer via assignment for all interlayer nets.

Thermally-driven 2D Maze Routing

After signal interlayer via assignment, we can decompose all interlayer nets into a set of 2D nets by adding a pseudo-pin at each signal interlayer via landing position. Moreover, with a known signal interlayer via distribution, we can perform temperature analysis as discussed in Section 4.4 and obtain an initial temperature map to guide 2D routing. 2D routing is performed on all nets using the maze routing algorithm from [6], which propagates wave-front-like expansion and connects all the pins of a net on a 2D layer. The cost function used in maze routing for taking a routing edge is similar to equation (4.14), but with an *additional temperature term* in the cost, so that the resulting route will have an incentive to choose a low temperature path. This strategy not only decreases the temperature influence on the delay of signal nets [10], but also reduces congestion in hot regions, which is beneficial, since heuristically more thermal vias will later be inserted in these regions to reduce temperature. The overflow will be resolved by iterative rip-up and reroute. After 2D routing, we can further connect the pins at different layers for those interlayer nets by employing the previously assigned signal interlayer vias, thus build an initial 3D routing solution.

Overall Flow of the 3D Global Routing Algorithm

We summarize the individual pieces of the algorithm and outline the overall flow in Figure 4.9. The input is a tessellated 3D circuit with given power distribution. An initial 3D routing

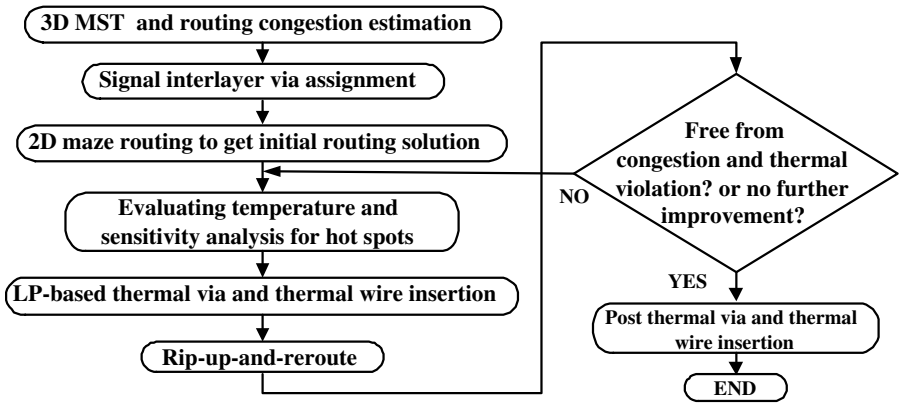


Figure 4.9: Overall flow for the temperature-aware 3D global routing algorithm.

solution is found by generating a 3D MST, and then performing a network flow based interlayer via assignment, followed by thermally-driven 2D maze routing. Based on the thermal profile and sensitivity information, we then perform a LP-based thermal via and thermal wire insertion procedure. This insertion is performed iteratively; each time after the insertion, a rip-up-and-reroute step is employed to resolve the lateral routing congestion and overflow. This continues until there is no temperature and congestion violation or no further improvement is possible.

The complexity of our algorithm is analyzed for each of the above steps, and we list them following the order of routing flow:

- The initial 3D MST decomposition and congestion estimation take $O(ND^2logD+NE)$ time, where N is the number of nets, D is the maximum number of pins for a net, and E is the number of lateral routing edges in the routing graph. The first term stands for the complexity of 3D MST decomposition, the second term is the complexity for congestion estimation.

- The interlayer signal via assignment models a min-cost network flow problem and solves it with the *cs2* package [46]. The complexity is $O(IG^2logG + (I^2G + IG^2)loglogR_{max}log((I + G)C_{max}))$, in which I is the number of interlayer nets, and G is the number of GRCs on each layer; R_{max} and C_{max} are, respectively, the maximum values of capacity and cost in the network flow graph. The first term represents the complexity to build the flow graph, and the second term is the time to solve the min-cost flow problem using the scaling algorithm [46].

- The iterative process of maze routing, thermal/sensitivity analysis and thermal via and thermal wire insertion has the following complexity. One iteration of maze routing has a complexity of $O(NGlogG)$ since it is an extension of Dijkstra's algorithm [35]. The temperature evaluation and sensitivity analysis involve the solution of a system of linear equations using the conjugate gradient method [104], and in practice, the cost of these computations is just over $O(n)$ for sparse and SPD equations as in our case, where n is the total number of nodes and thermal conductances in the thermal circuit; in our problem, n is of the same order as E, hence the complexity is just over $O(E)$. Let the linear programming problem in the thermal via/wire insertion step have d decision variables and c constraint, the simplex method used in LP_SOLVE [15] has a complexity of $O(d^2c)$ in practice. In the worst case, both d and c are of the same order as $v + E$, where v is the number of temperature violation spots. Therefore the complexity for our thermal via/wire insertion step is $O((v + E)^3)$.

- The rip-up-and-reroute, thermal/sensitivity analysis and thermal via/wire insertion are iteratively performed, and the majority of the algorithm run time is spent on executing this loop. However, it is seen in practice that the optimized solution can be reached in a small number of iterations.

4.6 Experimental Results

The temperature-aware 3D router was implemented in C++ and tested on an Intel Pentium 4 2.8GHz Linux machine, on benchmarks from the MCNC [81] and IBM placement benchmark [118] suites. Table 4.1 lists parameters of the benchmark circuits and routing grid size for each layer. The benchmarks are placed with the placer from [47]. Four layers are used, the chip size is fixed at 5mm $\times$ 5mm; the oxide layer separation is 7μm, and the silicon substrate is 500μm thick. The power of each circuit is randomly generated with a power density between 10W/cm^2 to 800W/cm^2. The white spaces of GRCs is calculated and the average is about 20% of the total chip area. The interlayer via has a size of 5μm $\times$ 5μm. The thermal conductivity of silicon, oxide and metal

are 119W/(m˚C), 1W/(m˚C) and 396W/(m˚C), respectively. The bottom of the chip was made isothermic with the ambient temperature to represent the heat sink, and the top and sides of the chip are assumed to be adiabatic to the ambient. The temperature of the heat sink is set to a reference value of 0˚C for convenience; if the temperature is nonzero, it is well known that the calculated values can be obtained by a simple translation.

Circuit	# cells	# nets	Grid	Circuit	# cells	# nets	Grid
biomed	6417	5743	28 ×28	ibm02	19321	18429	34×34
industry2	12149	12696	31×31	ibm03	22207	21905	37×37
industry3	15059	21939	35×35	ibm04	26633	26451	36×36
ibm01	12282	11754	31×31	ibm06	32185	33521	40×40

Table 4.1: Benchmark circuit parameters.

The experimental results are listed in Table 4.2. We compare our algorithm of temperature-aware 3D global routing (denoted as *TA*) with three other 3D global routing schemes. The first comparison scheme performs initial 3D global routing in the same way as our algorithm, but after that, both thermal vias and thermal wires are post-inserted into the 3D circuit in a greedy way, under the constraint of existing routing resource usage and white space allowance, and we denote it as *P* in Table 4.2. The second comparison scheme follows the same routing procedure as our approach, but only thermal vias are inserted and optimized to improve heat conduction; thermal wires are not used, and we denote this scheme as *V*. The third comparison scheme uses the same number of thermal vias and thermal wires as our routing algorithm, but they are distributed uniformly across the chip. The level of thermal via distribution is equal to the average number of thermal vias per GRC position from our algorithm, and the thermal wire level is assigned in the similar uniform way. We denote this comparison scheme as *U* in the table. For all four approaches, we set up our specified peak temperature T_{spec} to be 80˚C to guide the temperature reduction process.

Routing results and temperature performance for each approach are shown in Table 4.2, for a set of circuits listed in the first column. The second column T_{init} lists the peak temperature after simply performing initial 3D routing but without any thermal via and thermal wire insertion. As we can see, the peak temperature is well above our expected value without insertion of thermal

Circuit	T_{init} (°C)	Peak temperature (°C)				Ave. top 1% temperatures (°C)				Wire length ($\times 10^5$)				run time (seconds)			
		TA	*P*	*V*	*U*	*TA*	*P*	*V*	*U*	*TA*	*P*	*V*	*U*	*TA*	*P*	*V*	*U*
biomed	237.1	81.9	105.6	115.3	87.5	73.7	97.8	80.0	76.6	1.82	1.77	1.78	1.76	255	137	188	131
industry2	207.5	82.4	106.6	116.3	98.0	64.0	75.2	77.9	68.3	6.04	5.92	6.01	5.90	855	473	591	519
industry3	202.0	79.2	99.1	112.5	89.8	68.0	78.0	74.9	71.9	9.85	9.75	9.71	9.93	1807	1405	1686	1193
ibm01	264.8	79.1	109.9	99.3	108.4	61.7	86.9	68.7	74.4	2.63	2.46	2.63	2.56	238	87	224	87
ibm02	257.5	80.5	105.6	111.6	97.5	57.3	65.2	63.3	60.8	7.73	7.57	7.68	7.60	769	400	717	469
ibm03	218.5	82.9	106.9	123.4	85.4	68.1	78.1	77.8	67.3	10.27	10.30	10.26	10.00	1645	1490	1344	864
ibm04	218.1	80.0	96.0	107.8	84.6	68.6	73.3	78.1	68.7	8.07	8.15	8.21	8.25	886	597	974	581
ibm06	236.4	81.2	99.2	131.4	89.2	63.0	69.3	66.9	63.8	18.08	18.19	18.12	18.08	2956	1585	2274	1581

Table 4.2: Comparison of temperature and routing performance results among four approaches: 3D global routing using our thermal-aware (*TA*) method; using post (*P*) insertion of thermal vias and thermal wires; using thermal via (*V*) insertion only; and using uniform (*U*) thermal via and thermal wire insertion.

vias and thermal wires, which shows the necessity of these insertions in 3D ICs. The following four columns list the peak temperature for all routing algorithms. Experimental results show that our temperature-aware (*TA*) algorithm can successfully bring the peak temperatures of circuits down to specified temperature T_{spec} and performs much better than other routing schemes. The post-insertion of thermal vias and thermal wires in approach *P* does not optimize resource usage during routing phase and results in inferior performance; the peak temperature from *P* can be as high as 30.8°C more than that from *TA* algorithm (*ibm01*). Approach *V* does not employ thermal wires as an effective lateral heat conduction resource, thus for some spots where thermal via insertion is restricted from routing congestion, hot spots are generated due to the lack of lateral thermal conduction path; experimental results show a maximum peak temperature difference of 50.2°C (*ibm06*) between *V* and *TA*. The uniform insertion approach *U* applies thermal vias and wires uniformly instead of distributing them according to sensitivity analysis, therefore its performance is inferior to our *TA* routing algorithm. Moreover, it creates great routing overflow as discussed later. The next four columns listed the average temperature of top 1% hot spots for all four approaches. The results show the same tendency as peak temperature results, our temperature-aware algorithm can reduce the temperature of hot spots better than other approaches from an iterative routing and temperature reduction approach.

Table 4.2 also lists the wirelength and run time results. All four approaches report similar wirelength results. Averagely, our *TA* algorithm generates 0.6% longer wirelength than approach *P*, because during iterations *TA* algorithm will detour to leave space for thermal via and thermal wire insertion for temperature reduction and thus increase wirelength; however, this increase is trivial. *TA* and *V* algorithms need longer run time, which is due to the iterations to resolve temperature violation and routing congestion.

Circuit	# routing overflows				Circuit	# routing overflows			
	TA	*P*	*V*	*U*		*TA*	*P*	*V*	*U*
biomed	11	0	8	6750	*ibm02*	34	0	15	4262
industry2	1	0	4	6201	*ibm03*	11	0	1	16873
industry3	14	0	1	10021	*ibm04*	2	0	1	5596
ibm01	34	0	15	4262	*ibm06*	7	0	5	5404

Table 4.3: Comparison of routing overflow of four approaches.

Table 4.3 reports the lateral routing overflow of four algorithms. The *TA* algorithm can successfully resolve routing congestion and temperature violations by employing LP-based thermal via and thermal wire insertion as well as iterative approach, and reports trivial routing overflow. However, the uniform insertion algorithm *U* distributes thermal vias and thermal wires uniformly across the chip without addressing the routing capacity reduction from them, thus approach *U* generates huge routing overflow, which makes the routing infeasible. Overall, the experimental results show that our temperature-aware algorithm can effectively reduce hot spot temperature in 3D ICs to specified value, and maintain a good wirelength and overflow property through an iterative LP-based thermal via and thermal wire insertion as well as rip-up-and-rerouting.

4.7 Conclusion

In this chapter, we have proposed an effective temperature-aware global router for 3D ICs. The 3D global router employs procedures for heat conduction using both thermal vias and thermal wires to reduce the chip temperature to the desired value. Our algorithm utilizes sensitivity analysis and an LP-based approach to resolve the contention between thermal vias, thermal wires and lateral routing congestion in an iterative way. Experimental results show that our 3D global routing algorithm can efficiently bring down the circuit temperature to desired level, and finds a routable solution with good wirelength characteristics.

Chapter 5

Buffering Global Interconnects in Structured ASIC Design

Structured ASICs present an attractive alternative to reducing design costs and turnaround times in nanometer designs. As with conventional ASICs, such designs require global wires to be buffered. However, via-programmable designs must prefabricate and preplace buffers in the layout. This chapter proposes a novel and accurate statistical estimation technique for distributing prefabricated buffers through a layout. It employs Rent's rule to estimate the buffer distribution required for the layout, so that an appropriate structured ASIC may be selected for the design. Experimental results show that the buffer distribution estimation is accurate and economic, and that a uniform buffer distribution can maintain a high degree of regularity in design and shows a good timing performance, comparable with nonuniform buffer distribution.

5.1 Introduction

In the nanometer regime, cell-based ASIC designs are increasingly hard-pressed to produce affordable design solutions, due to the challenges associated with skyrocketing mask costs and manufacturability issues for complex designs [49]. As an alternative, field programmable gate

arrays (FPGAs) may be used since they provide a regular prefabricated structure that can avoid many of the manufacturing problems associated with ASICs. However, for many designs, the performance gaps, in terms of speed, power, and area, between FPGAs and cell-based ASIC designs are too large for an FPGA to be a realistic alternative. In this context, structured ASIC design has emerged as a promising new design style to fill the gap.

Structured ASICs are composed of regular arrays of prefabricated standard building blocks, with fixed mask structures. Design with structured ASICs involves many fewer masks than for cell-based ASICs. One paradigm that is used involves *via-configurability*, where the building blocks and interconnect skeletons are prefabricated and are then connected with appropriate via connections by programming only a small number of masks [59, 70, 91, 94, 96]. A standard building block is composed of combinational logic and memory elements (such as flip-flops) and has enough flexibility that it can be programmed to various functions through via configurations. The wires that electrically connect the building blocks are also prefabricated regular fabrics, and the routing of nets can be configured with a via definition. This strategy provides a low NRE (non-recurring engineering) cost, and yet with relatively high-performance solutions. Additionally, since structured ASICs use well-characterized logic blocks and regular, fixed interconnect structures, they are well suited to combat problems associated with manufacturability, yield, noise, or crosstalk.

However, many of the other problems of nanometer design continue to plague structured ASICs. Most importantly, as in cell-based ASICs, the dominant role of interconnect in determining system performance remains a major hurdle, and a key issue in overcoming this is through the use of buffer insertion along global wires. As discussed in Section 2.3, buffer insertion can not only improve timing performance, but can also effectively reduce functional noise by recovering noise margins. The number of buffers necessary to achieve timing closure and meet noise requirements continues to rise with decreasing feature sizes. Although interconnects are generally buffered in the later phases of physical design, buffer resources must be planned earlier in the design process, so that enough resources are available for the later insertion phase. Section 2.3 discusses several buffer resource planning strategies for cell-based ASIC and custom design.

This problem is more acute for via-programmable structured ASICs, where, in addition to the basic standard building blocks, buffers must be prefabricated and distributed in the layout. Although it is possible to reconfigure the basic building blocks to work as buffers, this is not an economical approach since (a) the number of buffers can be very large, (b) the sizes of the buffers are typically larger than the sizes of regular gates, (c) configuring a large and general standard building blocks as buffers is an inefficient use of resources, and (d) these blocks do not have the driving ability of dedicated buffers. Therefore, it is essential to distribute dedicated buffers in structured ASICs, and to plan for them well, prior to the fabrication of the chip.

The buffer insertion problem for structured ASIC design has not been fully addressed in publications so far. The only work we know of that considers this issue is [114], where it is assumed that a uniform distribution of dedicated buffers is placed throughout the layout, and there is a ratio of 2:1 between number of logic cells and buffers everywhere in the circuit. However, this does not recognize that the demand for buffers depends on the interconnect complexity of circuits, and assuming a single 2:1 ratio for all kind of circuits may result in a large waste of buffer resources.

Clearly, the choice of this ratio should depend on the topology and structure of the circuit that is being mapped to the chip and the number of interconnects. On the other hand, given that this ratio must be predetermined in a structured ASIC, it is clearly not possible or realistic to tune each chip individually to a design, and a "good" set of buffer-to-logic-cell ratios must be chosen.

This chapter uses Rent's rule, which we have presented in preliminary Section 2.5, to develop a family of "good" ratios, and proposes a distributed buffer insertion methodology for the use of dedicated buffers in structured ASIC design. For each range of (p, k) values, where p is the Rent's exponent and k the Rent's coefficient, our algorithm (described in Section 5.3.2) finds a statistical estimate of the buffer distribution for circuits falling in that range. Thus for each range, we can prefabricate an off-the-shelf structured ASIC chip with buffers preplaced according to the estimated buffer distribution of that (p, k) range.

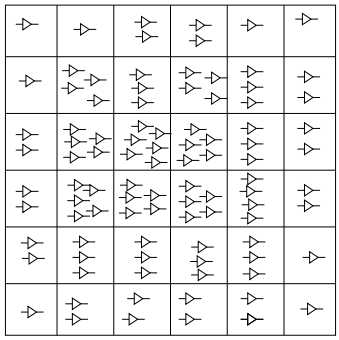
In the implementation phase, a designer may choose the appropriate prefabricated chip for implementing custom design according to values of (p, k) for the design. Experimental results show

that the buffer resource estimation is accurate and adequate for interconnect buffering purpose of circuits in each range, and with an average uniform buffer distribution based on the estimation, we can maintain a good timing performance as well as a highly regular structured ASIC.

The organization of the chapter is as follows. In Section 5.2, we describe the buffer insertion methodology. Section 5.3 provides a statistical buffer distribution estimation based on Rent's rule, and a classification method of circuits based on their Rent's exponent and coefficient. In Section 5.4, we present the experimental results, which is followed by conclusion in Section 5.5.

5.2 Buffer Insertion Scheme for Structured ASIC Design

As with other functional units, buffers must be prefabricated and distributed in structured ASICs. A distributed buffer model for ASIC and custom design is discussed in Section 2.3, which bears the virtues of increased flexibility and reduced congestion compared with buffer block planning. In the same spirit of interspersing buffers with logic units across the circuit as in ASIC and custom design, we adopt a buffer placement model in which the prefabricated buffers are scattered through the structured ASIC, and the distribution of buffers should be adequate enough for buffering global wires. We also refer to this as a distributed buffer model to capture the distributed nature of this scheme, although unlike in ASIC and custom design, instead of allocating buffer sites, buffers are actually prefabricated in structured ASICs.



(a)

1	1	2	2	1	1
1	4	3	4	3	2
2	5	6	5	3	2
2	5	5	5	4	2
2	3	3	3	3	1
1	2	2	2	2	1

(b)

Figure 5.1: (a) A schematic showing a chip that is tessellated into tiles, with buffers dispersed within the tiles. For simplicity, the VCCs are not shown here. (b) The corresponding tessellation with the buffer capacity listed for each tile.

Structurally, we define the structured ASIC to be a two dimensional array composed of via-configurable standard building blocks, denoted as *via-configurable cells* (VCC), which are connected by via-configurable interconnect wires. Similar to the tessellation of ASIC and custom design in Section 2.3, to facilitate the distributed buffering model in structured ASIC, we divide the circuit into an array of *tiles*, and each tile is a square area containing $m \times m$ VCCs as well as a predetermined number of dedicated buffers for interconnect buffering usage. For a tile positioned at (i, j), we refer to this number as the *buffer capacity*, denoted as $B_{i,j}$. If $B_{i,j}$ is uniform for all (i, j), we refer to this as a *uniform* buffer distribution; otherwise the buffer distribution is said to be *nonuniform*. A routing solution may use some or all of these prefabricated buffers: the actual number of utilized buffers in tile (i, j) is referred to as the *buffer usage*, denoted as $b_{i,j}$. If $b_{i,j} > B_{i,j}$, the routing solution is invalid, and we refer to this situation as *buffer overflow*. Figure 5.1(a) shows a 6×6 tile graph for a structured ASIC design, with buffers distributed within tiles. The corresponding buffer capacity of each tile in the structured ASIC with a nonuniform buffer distribution is shown in Figure 5.1(b).

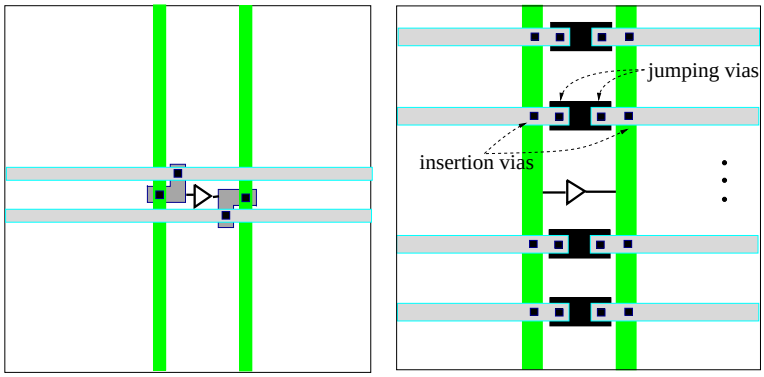


Figure 5.2: (a) The input and output of a buffer are each connected to two wires that stretch across the tile, one in the horizontal direction and one in the vertical direction. (b) Buffer insertion on a wire crossing the tile is carried out by means of via definition. An "insertion via" connects a buffer, while a "jumping via" is an electrical short circuit.

In this paradigm, buffers are prefabricated into the layout but are connected to the global lines that require buffering only in the later phases of physical design. To enable this, we utilize a via-defined buffer insertion (VDBI) scheme, which allows a buffer in a tile to be inserted along any interconnect wire traversing the tile, by means of a via configuration. Figure 5.2(a) shows

a representative buffer in a tile: its input and output are connected to horizontal and vertical wires that go across the tile. Figure 5.2(b) shows that any wire that crosses the tile can choose to be via-configurably connected to either this buffer through "insertion vias," or through "jumping vias" to a metal strip that can skip the buffer entirely.

Similar to the case of buffering ASIC and custom design as discussed in Section 2.3, our buffer insertion model uses the same simple yet effective distance-based criterion for structured ASICs: the maximum total length of interconnect can be driven by a gate (buffer) is L, and this is referred to as the *critical length*. This simple buffer insertion constraint is effective and flexible enough and can also work with various routing algorithms.

5.3 Statistical Buffer Distribution Estimation

Structured ASICs provide a reliable and economic platform to implement various designs, but the trade-off is the reduction in flexibility. The basic VCCs, buffers and interconnects must be prefabricated, and this necessitates early and accurate estimation of physical design properties without knowing the details of circuit. To ensure that adequate numbers of buffers are available for routing the circuit, we must determine the buffer distribution, i.e., the buffer capacity of each tile, prior to fabrication. This estimate of the buffer distribution should have the following properties:

1. It is desirable for this estimate to be accurate, and should guarantee that enough buffers will be allocated so that it can meet the timing-optimal buffer usage in the final layout. On the other hand, the estimate can not be too pessimistic: if too many buffers are prefabricated, it will result in wasted silicon area.

2. The estimation should be based on basic circuit properties instead of any specific circuit implementation details, so that a set of prefabricated chips can be developed on the basis of these properties. Since these chips will be used to implement a variety of designs, and may work with various physical design tools, the solution should not be specific to any particular tools, and should only be based on basic circuit properties.

With the above considerations, our buffer distribution estimation is based on Rent's rule which is detailed described in Section 2.5. We use this to determine a statistical estimate of the interconnect wire distribution, and the buffer distribution. In the remainder of this section, we will describe the background and details of our estimation technique.

5.3.1 Estimating the Statistics of Buffer Distribution

The number of buffers required in a tile is highly correlated to the total length of external interconnect wires crossing this tile. If a larger number of wires traverse a given tile, it is likely that a larger number of buffers will have to be inserted in the tile. From the Rent's exponent p and Rent's coefficient k for a circuit, we can apply Rent's rule to statistically estimate the length of interconnect wires crossing a specific tile D, and further, to estimate the number of buffers required in the tile.

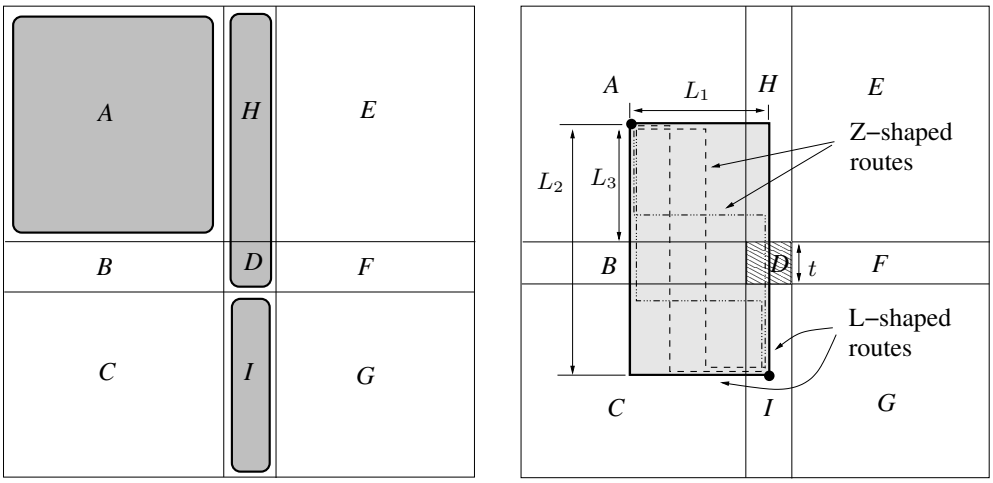


Figure 5.3: (a) We divide the circuit into 9 blocks, A through I, and for estimation purpose, we merge two blocks H and D into a larger block HD. (b) Estimation of the length of interconnect wires passing tile D.

A schematic of a circuit layout is shown in Figure 5.3(a). The layout is a square consisting of $n \times n$ tiles[1]. Each tile is a square consisting of $m \times m$ VCCs, and the geometrical size of a tile is $t \times t$ units. While considering the estimation for a tile D, we divide the circuit, for convenience, into

[1]For a general circuit of rectangular form, the analysis is very similar to the square case.

9 blocks, labeled A through I, as shown in the figure, with block D in the center. Block A consists of all tiles northwest of D; block B is composed of all tiles to the east of E; block C comprises the set of tiles to the southwest of D, and so on. We will use (i, j) to refer to the coordinates of tile D in the $n \times n$ tiling.

For estimation purposes, it is reasonable to assume that the routing will remain within the bounding box set by the pins of a net. Under this assumption, the interconnect wires that cross block D consist of the contributions of the wires connecting block pairs in set S defined as

$$\begin{aligned} S \quad = \quad & \{(A,F),(A,G),(A,I),(B,E),(B,F),(B,G),(B,H),(B,I),(C,E), \\ & (C,F),(B,H),(B,I),(C,E),(C,F),(C,H),(E,I),(F,H),(F,I), \\ & (G,H),(H,I)\} \end{aligned}$$

The total wire length passing through tile D, W_D, is given by

$$W_D = \sum_{all\ block\ pairs} \sum_{(x,y)\in S} W_{x,y}, \tag{5.1}$$

where $W_{x,y}$ is the total wirelength of all interconnects crossing tile D, connecting VCCs in blocks x and y, where $(x, y) \in S$. Since the tile dimension t is generally much smaller[2] than the critical length L, the interconnects originating from tile D will not be likely to consume buffer resources at D, and we only consider the contribution from those wires *passing* D. Knowing W_D, if the maximum interconnect length driven by any buffer is length L units from the insertion model described in Section 2.3, then a *unit length* interconnect segment in a wire crossing tile D will have a probability of $1/L$ of requiring a buffer to be inserted in tile D. This implies that an external wire passing a tile of dimension t horizontally will probabilistically insert t/L buffers from this tile. We can use this idea to estimate the buffer capacity, B_D, required for tile D as the probabilistic

[2]It is easy to extend this analysis to the case where t is larger than L.

usage of buffers in the tile. In other words,

$$B_D = \frac{W_D}{L} \tag{5.2}$$

The $W_{x,y}$ components of W_D can be estimated by using Rent's rule. As an example, we now illustrate how the value of $W_{A,I}$ may be estimated; other $W_{x,y}$ components may be estimated in a similar way. As in [38], we merge two neighboring blocks H and D into a larger block HD as shown in Figure 5.3(a), and apply the I/O terminal conservation rules to the three blocks A, HD and I, which are shown as shaded regions in Figure 5.3(a). We have the number of I/Os connecting blocks A and I, denoted as $T_{A\_to\_I}$, to be

$$T_{A\_to\_I} = T_{AHD} + T_{HDI} - T_{HD} - T_{AHDI} \tag{5.3}$$

in which the $T_{block}, block \in \{AHD, HDI, HD, AHDI\}$ is the number of I/Os of the combinational blocks, and they can be estimated using Rent's rule as

$$\begin{aligned} T_{AHD} &= k(N_A + N_H + N_D)^p \\ T_{HDI} &= k(N_H + N_D + N_I)^p \\ T_{HD} &= k(N_H + N_D)^p \\ T_{AHDI} &= k(N_A + N_H + N_D + N_I)^p \end{aligned} \tag{5.4}$$

The parameters N_A, N_H, N_D and N_I are the number of VCCs in blocks A, H, D and I, and they are simple expressions in the variables i, j, n and m. However, in applying this formula, we may find that the estimate moves into region II of Rent's rule as described in Section 2.5. We use a simplified approach to handle this deviation: when the number of VCCs in the combinational block from equation (5.4) exceeds some critical size N_{crit}, we substitute this number with N_{crit} into equation (5.4). Experimental curves show that N_{crit} is between 150 to 200 for various circuits, and we simplify it by taking $N_{crit} = 175$ for all experimental circuits. Experimental results also

show that small variation in the choice of N_{crit} does not affect estimation results much.

To calculate the number of interconnects between blocks A and I, we define a variable α that is the fraction of terminals that are sinks. Thus we can obtain the number of point-to-point interconnects between block A and block I, $I_{A\_to\_I}$ as:

$$I_{A\_to\_I} = \alpha T_{A\_to\_I} \tag{5.5}$$

and α can be expressed as the average fanout of the system, as

$$\alpha = \frac{fanout}{fanout + 1} \tag{5.6}$$

Using equation (5.5) to obtain the number of interconnects between blocks A and I, we can further combine it with a simplified L-Z shaped routing model to estimate the wire length crossing tile D due to interconnects between A and I, $W_{A,I}$. Figure 5.3(b) shows a set of possible L-shaped and Z-shaped connections between the blocks A and I. Probabilistically, we can assume that the average position of the terminals of the interconnects are at the center of blocks A and I. Thus the routing of interconnects will follow the bounding-box path, and falls in the dotted box in Figure 5.3(b). We denote the distance between the centers of A and H by L_1; the distance between the centers of A and C by L_2; and the distance between the center of A and the northern edge of D as L_3. The parameters L_1, L_2 and L_3 are pictorially illustrated in Figure 5.3, and these can be expressed in terms of i, j, n and t.

In practice, it is observed that a router will route the bulk of the nets with simple L-shaped and Z-shaped patterns [113]. Hence, we can assume that the routing of an interconnect will utilize one of these two patterns, and the probability of using an L-shaped and Z-shaped route are P_L and $P_Z = 1 - P_L$, respectively. As in [113], we assume $P_L = 0.7$ in the estimation. Under this routing model, we can estimate the wirelength crossing tile D due to L-shaped routes as:

$$W_{L,(A,I)} = \frac{1}{2} \cdot t \cdot I_{A\_to\_I} \cdot P_L \tag{5.7}$$

The factor "1/2" in the above equation is due to the fact that there are two possible L-shaped routes, and only the upper-L route will pass the D tile.

Similarly, there are two kinds of Z-shaped routes, type I: with two horizontal segments, and type II: with two vertical segments. If every Z-shaped route has an equal probability of being taken, the type I Z-shaped routes will have a probability of $L_1/(L_1+L_2)$ of being taken, while the type II Z-shaped routes have a probability of $L_2/(L_1+L_2)$. Both types of routes can pass tile D, and we can probabilistically estimate the wirelength of Z-shaped routes crossing tile D as:

$$W_{Z,(A,I)} = \left(\frac{L_1}{L_1+L_2} \cdot \frac{t/2}{L_1} \cdot t + \frac{L_2}{L_1+L_2} \cdot \frac{(L_3+t)}{L_2} \cdot t \right) \cdot I_{A\_to\_I} \cdot (1-P_L) \tag{5.8}$$

Finally, we can compute the total interconnect wire length from A to I that traverses D as

$$W_{A,I} = W_{L,(A,I)} + W_{Z,(A,I)} \tag{5.9}$$

The wirelength contribution from other block pairs in set S can be computed in a similar way as $W_{A,I}$, and their values can be substituted into equation (5.1) and (5.2) to yield the estimated buffer capacity B_D for tile D at position (i,j).

5.3.2 Application to Structured ASICs

Using (p,k) Range to Prefabricate Structured ASIC Templates

Structured ASICs consist of predetermined regular architectures, and the buffer resource planning must be addressed at the prefabrication phase. However, due to the unavailability of specific circuit information at this phase, we must preallocate buffers so that it can satisfy the buffer insertion requirements of a range of circuits. As stated earlier, our approach determines the buffer distribution that is necessary for a circuit, based on its basic characteristics, namely, the Rent's exponent p and Rent's coefficient k. The practical way of employing structured ASICs in design is that they are prefabricated with other hard Intellectual Property (IP) blocks, which are embedded processors, I/O controllers, etc. The structured ASIC part can be used to implement

users' specific logic, and we can assume that there is only one Rent's exponent p and one Rent's coefficient k associated with this logic.

We now describe how the buffer distribution determined in section 5.3.1 is used to design off-the-shelf structured ASIC parts that can be used for a specific circuit. We can divide the spectrum of Rent's exponent values, p, and Rent's coefficient values, k, into a set of ranges: $R_p = \{[p_1, p_2]$, $[p_2, p_3]$, $[p_3, p_4], ...\}$, $R_k = \{[k_1, k_2]$, $[k_2, k_3]$, $[k_3, k_4]$, $...\}$. For the circuits of tile array size $n \times n$ in a specific range pair $[p_i, p_{i+1}]$ and $[k_j, k_{j+1}]$, we can predetermine the maximum number of buffers required in each tile for these circuits with the algorithm shown in Figure 5.4. Using this estimation technique, a set of structured ASIC chips can be prefabricated. When a given circuit is to be mapped onto this fabric, its Rent's parameters are first computed. Based on their values, the appropriate prefabricated structured ASIC chip is chosen, and the circuit is mapped onto that chip. If the Rent's parameter is exactly at the boundary between two ranges, we choose the structured ASIC representing the lower range to avoid waste of buffer resource.

To find a buffer distribution fitting the requirements of all circuits in the range $[k_j, k_{j+1}]$ and $[p_i, p_{i+1}]$, we must find the maximum estimated buffers in each tile. The estimation procedure in Figure 5.4 is based on the following theorem and observation:

Theorem 1. *The estimated number of buffers in a tile D is a monotonically increasing function of the Rent's coefficient, k.*

Proof. According to the statistical estimation in Section 5.3.1, the estimated number of buffers B_D in a tile D has a general expression as follows:

$$B_D = \sum_{all\ block\ pairs\ i \in\ S_D} \eta_i \cdot k[(N_{\alpha,i} + N_{\beta,i})^p + (N_{\beta,i} + N_{\gamma,i})^p - (N_{\alpha,i} + N_{\beta,i} + N_{\gamma,i})^p - N_{\beta,i}^p] \tag{5.10}$$

where S_D is the set of block pairs, interconnects between which can pass tile D; η_i is a constant determined for block pair i; $N_{\alpha,i}$, $N_{\beta,i}$ and $N_{\gamma,i}$ are the number of VCCs in three consecutive neighboring blocks α_i, β_i and γ_i. From this expression, we can conclude that the estimated

Algorithm: `Estimate_buffer_distribution_for_circuits_in_a_range`

Input: Tile array of size $n \times n$; range of Rent's exponent $[p_i, p_{i+1}]$; range of Rent's coefficient $[k_j, k_{j+1}]$.
Output: A buffer capacity estimation that can meet the buffer insertion requirements of all circuits falling in the above ranges.

```
begin
1.    k = k_{j+1};
2.    for each tile D in the tile array
3.        max_num_buffer(D) = 0;
4.        for p = p_i; p ≤ p_{i+1}; p = p + δ
5.            Estimate buffer usage at D, i.e., num_buffer(D), using the method in
              section 5.3.1 with Rent's exponent and coefficient (p, k);
6.            if (max_num_buffer(D) < num_buffer(D))
7.                max_num_buffer(D) = num_buffer(D);
8.        end for
9.        set buffer capacity at D to be max_num_buffer(D);
10.   end for
end
```

Figure 5.4: Estimation of buffer distribution for a range of circuits.

number of buffers in a tile D is linearly dependent on k, hence it is a monotonically increasing function of k. □

Observation 1. *The estimated number of buffers in a tile D in general does not vary monotonically with Rent's exponent p.*

This observation can be illustrated as follows. From equation (5.10) above, the derivative of the estimated number buffers B_D in tile D, with respect to Rent's exponent, p, can be found as:

$$\begin{aligned} \mathrm{d}B_D/\mathrm{d}p &= \sum_{all\ block\ pairs\ i \in S_D} \eta_i \cdot k[log(N_{\alpha,i} + N_{\beta,i}) \cdot (N_{\alpha,i} + N_{\beta,i})^p \\ &+ log(N_{\beta,i} + N_{\gamma,i}) \cdot (N_{\beta,i} + N_{\gamma,i})^p - log(N_{\alpha,i} + N_{\beta,i} + N_{\gamma,i}) \\ &\cdot (N_{\alpha,i} + N_{\beta,i} + N_{\gamma,i})^p - logN_{\beta,i} \cdot N_{\beta,i}^p] \end{aligned}$$

With the values of $N_{\alpha,i}$, $N_{\beta,i}$, $N_{\gamma,i}$, η_i and p varying, this derivative can be of positive or negative value. Therefore, the estimated number of buffers in a tile D generally does not vary monotonically with Rent's exponent p.

Based on the above theorem and observation, we design the algorithm in Figure 5.4 to estimate the maximum number of buffers required in each tile. According to Theorem 1, line 1

sets k to be at the upper limit of the range, i.e., k_{j+1} so as to find the maximum buffers in each tile. However, with Observation 1, the dependence between the estimated number of buffers and p is not monotonic, and therefore, line 3 through 9 performs an enumeration of p with a step size of δ to find the maximum number of estimated buffers in a tile D for that range. In our experiments, we find $\delta = 0.01$ is an appropriate choice of the step size. Finally, this enumeration is embedded in an outer loop (line 2 through 10) to perform such estimation for all tiles.

With this estimated buffer capacity for circuits that lie within a range of Rent's parameter values, we can predetermine a single buffer distribution to satisfy the interconnect buffering requirement of all of these circuits, thus creating only one structured ASIC chip that can meet the requirements of all of these circuits. The finer the granularity of these ranges, the more accurate the buffer distribution will be, but the trade-off is that a larger number of base chips will have to be prefabricated. The size of the structured ASICs is determined at the prefabrication phase and it may be larger than the circuit to be implemented. However, for the same (p, k) range, it can be shown that the estimated buffer capacity for a larger prefabricated structured ASIC will be greater than the estimated buffer usage for a smaller circuit, if we place the smaller circuit at the center of the structured ASIC. Thus if we estimate and distribute buffers aiming at a larger base structured ASIC, it will always satisfy the buffering requirement of smaller implemented circuits.

Uniform Buffer Distribution

The buffer distribution estimation obtained above is not a uniform one, i.e., the number of buffers at different tile locations are different. An important feature of structured ASICs is the regularity in design, which helps much in improving the manufacturability and reducing the design complexity. If a single buffer-to-logic-cell ratio is chosen, a single tile can be designed and optimized, and then repeated throughout the layout. In other words, an alternative to the above *nonuniform* estimated buffer distribution is a *uniform* buffer distribution. We set the number of buffers each tile in the uniform distribution as the *average* buffer number over all the tiles in the nonuniform distribution, which is obtained from the algorithm in Figure 5.4. This uniform

buffer distribution allocates the same amount of total buffers as the nonuniform distribution in the prefabrication phase, and more importantly, it maintains the regularity of the structured ASICs. Since its buffer level is based on the average buffer number from our estimation, it could still satisfy the requirement of interconnect buffering under our flexible buffer insertion model, and this is verified in experimental part.

5.4 Experimental Results

Our experiments are performed on 18 of the largest MCNC benchmarks, ranging from 1047 to 8383 logic blocks [16]. These circuits have been technology-mapped to 4-LUTs and flip flops using Flowmap [29], and then the 4-LUTs and flip flops are combined into basic logic blocks with VPACK tools [17]. The benchmarks are then placed and routed with Versatile Place and Route (VPR) tools [17] under a $0.09\mu m$ technology. In placement and routing, the VCC block used is the same as the VPGA based 4-LUT CLB designed in [91], and the routing architecture is the switch block architecture also from [91]. The technology parameters of $0.09\mu m$ technology used in physical design are listed in Table 5.1, and they are derived from the scaling of parameters in [91] as well as from [27] and [7].

Parameter	Value
VCC intrinsic delay (ps)	90
VCC area (μm^2)	52
VCC input capacitance (fF)	6.0
VCC output resistance (Ω)	916
Unit length wire resistance (Ω/μm)	0.143
Unit length wire capacitance (fF/μm)	0.25
Buffer input capacitance (fF)	13.65
Buffer output resistance (Ω)	231
Buffer intrinsic delay (ps)	28

Table 5.1: Technology parameters used in placement and routing.

We take each tile to include $8 \times 8 = 64$ VCCs, and compute the tile array size for each circuit, which is listed in Table 5.2. For the buffer-related parameters, the critical length L for buffer insertion is estimated using the similar approach from [99], which results in $L \approx 500\mu$m. We divide the Rent's exponent and coefficient spectrum into range sets $R_p = \{[0.3, 0.4], [0.4, 0.5],$

Circuit	p	k	Tile array	Circuit	p	k	Tile array
alu4	0.628	4.38	5 × 5	*frisc*	0.695	4.39	7 × 7
apex2	0.640	4.28	5 × 5	*misex3*	0.628	4.38	5 × 5
apex4	0.657	4.23	4 × 4	*pdc*	0.674	3.93	8 × 8
clma	0.578	4.37	12 × 12	*s298*	0.528	4.28	6 × 6
des	0.389	4.34	8 × 8	*s38417*	0.437	4.15	10 × 10
diffeq	0.460	4.07	5 × 5	*s38584.1*	0.413	4.18	10 × 10
elliptic	0.641	3.87	8 × 8	*seq*	0.616	3.98	5 × 5
ex1010	0.573	4.49	8 × 8	*spla*	0.676	3.99	8 × 8
ex5p	0.675	4.39	4 × 4	*tseng*	0.496	3.94	4 × 4

Table 5.2: Basic information about circuits: Rent's exponent, Rent's coefficient and tile array size.

$[0.5, 0.6]$,$[0.6, 0.7]\}$, and $R_k = \{[3.0, 4.0]$, $[4.0, 5.0]\}$, corresponding to a total of $|R_p| \times |R_n| = 8$ different types of structured ASICs to be fabricated. For each benchmark circuit, we can derive the Rent's exponent p and coefficient k in a recursive partitioning process using hMetis [65] and they are listed in Table 5.2. Once this is calculated, we then select the corresponding range in R_p and R_k for each circuit. Assuming the prefabricated base structured ASIC has similar tile array size as the actual circuit implementation, we can apply the estimation algorithm in Figure 5.4 to find the *estimated* number of buffers in each tile. We use this as the estimated *nonuniform* buffer capacity for the prefabricated structured ASIC characterized by the corresponding ranges in R_p and R_k, and we denote this as nonuniform (NU) distribution. Based on this, we can further calculate the average number of buffers per tile as the level of the estimated *uniform* buffer capacity, and we denote this as uniform (UNI) buffer distribution. We also experiment on a buffer distribution that the ratio between logic cells and buffers is 2:1 everywhere in the structured ASIC, and we denote this as a uniform 2:1 ($U21$) distribution. In our experimental setup, there are 32 buffers in each tile for the $U21$ distribution as 64 VCCs are contained in each tile. We then compare the accuracy and the timing performance of the three buffer distribution models.

To verify the choices that have been made, we apply a buffer insertion algorithm from [5] to actually insert buffers into the above placed and routed circuits, under the NU, UNI and $U21$ buffer capacity models. This will produce the *actual* number of buffers used in each tile. Comparing this actual buffer number distribution with that from the estimation models, we can examine the accuracy of our buffer distribution estimation, and the performance of three buffer distribution

Circuit	Ave. # buffers per tile				# buffer overflow per tile			Ave. buffer usage rate			Critical path delay (ns)		
	EBC	NU	UNI	$U21$	NU	UNI	$U21$	NU	UNI	$U21$	NU	UNI	$U21$
alu4	11	9.48	9.52	9.24	0	0.2	0	85%	86%	29%	2.02	1.91	2.00
apex2	13	9.60	9.44	9.20	0	0	0	74%	73%	29%	1.80	1.90	1.83
apex4	10	8.06	8.19	7.75	0	0.13	0	85%	86%	24%	2.04	1.95	2.05
clma	14	10.62	11.15	10.54	0	0.6	0	76%	79%	33%	3.35	3.33	3.30
des	4	3.02	3.33	2.90	0	0.72	0	79%	88%	9%	1.54	1.61	1.51
diffeq	4	3.64	3.60	3.48	0.08	0.32	0	88%	87%	11%	2.12	2.43	2.11
elliptic	10	8.19	8.53	8.30	0	0.44	0	81%	85%	26%	3.55	3.95	3.62
ex1010	11	7.81	8.06	7.58	0.02	0.08	0	69%	71%	24%	2.31	2.50	2.29
ex5p	8	6.25	6.81	6.69	0	0.13	0	83%	90%	21%	2.02	1.88	1.84
frisc	16	11.92	13.71	12.22	0.02	0.73	0	76%	87%	38%	4.19	4.59	4.27
misex3	7	5.40	4.92	5.28	0	0	0	78%	71%	17%	1.45	1.68	1.47
pdc	14	11.38	11.28	11.26	0	0.09	0	83%	82%	35%	2.46	2.55	2.48
s298	8	5.50	5.58	5.50	0	0	0	69%	70%	17%	4.15	4.43	4.16
s38417	7	5.81	5.54	5.26	0.01	0.27	0	82%	83%	16%	3.03	3.71	3.06
s38584.1	7	4.06	4.62	3.99	0.01	0.05	0	62%	69%	13%	2.70	2.99	2.58
seq	9	8.36	8.08	8.20	0	0.32	0	90%	87%	26%	1.87	1.97	1.76
spla	12	7.69	7.83	7.81	0	0.03	0	64%	62%	24%	2.03	2.30	2.05
tseng	3	2.44	2.19	2.31	0	0	0	81%	73%	7%	2.08	2.08	2.09

Table 5.3: Comparison of the buffer usage and timing performance results from buffer insertion under three buffer distribution models: nonuniform (NU) distribution, uniform (UNI) distribution and uniform 2:1 ($U21$) distribution. EBC is the estimated buffer capacity from our estimation algorithm.

models. The results of experiments are listed in Table 5.3. The first column of Table 5.3 lists the circuit names. The next four columns list the average number of buffers per tile for the estimated buffer capacity (EBC), as well as the actual buffer usages from buffer insertion under the three different buffer distribution models. The value of EBC has been rounded to the nearest integer, and it is exactly the uniform buffer capacity used in the uniform (UNI) distribution model. The uniform buffer capacity for the $U21$ model is 32 for all structured ASICs, and is not explicitly listed in the table. As we can see, the average number of buffers inserted in the cases of NU and UNI models are close to those computed from our statistical estimation; in fact, the actual number of buffers required is always a little smaller than the estimated value. This is due to the fact that our buffer estimation method determines the maximum number of buffers required for a set of circuits falling in a range of Rent's exponent and coefficient, and therefore, for a specific circuit with a particular set of Rent's parameters, this estimation can be larger than its actual usage. On the other hand, this pessimism also increase the robustness of the estimated capacity, so that the estimated buffer capacity will satisfy the buffering requirement of circuits even in the presence of fluctuations in the buffer usages of practical circuits. For the average buffer usage under $U21$ model, the average number of buffers used is much less than the capacity, 32, and this suggests an inefficient use of buffer resources in all of the benchmark circuits.

The columns labeled "# buffer overflow per tile" report the average buffer usage overflow per tile for each of the three buffer distribution models. $U21$ model, due to its extreme pessimism, results in no overflow at all, but at the price of large waste of resources. For the NU model, it is observed that most of the circuits are free from buffer overflow, and the occurrence of instances that do have overflows is very low, less than 0.08 per tile. The UNI buffer distribution model results in more overflows than the NU model, because it is derived from the UNI model, but the uniformity of the distribution incurs overflow, and it can be considered as a trade-off for the regularity in buffer distribution. However, it is observed these overflows are of low values, and even the worst case has less than 0.73 overflow per tile. These results show that our buffer estimation method can produce an adequate solution, so that the solution can successfully satisfy the buffer

insertion requirements of practical circuits. Moreover, even at the trade-off of regularity, our uniform distribution estimation still shows good estimation of buffer levels. In practice, to further reduce the buffer usage overflow under the UNI buffer distribution model, we can introduce a "fudge factor"to inflate the EBC value used in the UNI model. We set the inflated EBC value $EBC_{inf} = (1 + \epsilon)EBC$, thus increase the uniform buffer number to compensate for the trade-off resulted from the regularity. In experiments, we find that by choosing $\epsilon = 0.2$, most of the benchmark circuits will be overflow-free, and only three of them still have trivial overflow of less than 0.1 per tile.

The next three columns list the average buffer usage rate of all circuits under different buffer distribution models. It shows that for most of the circuits, the average buffer usage rate is between 70%-90% for both NU and UNI models, which means that the pessimism of our estimation algorithm is low, and that it actually produces reliable and economical *a priori* buffer distribution solutions. However, due to the extreme high buffer capacity, the $U21$ buffer distribution shows a very poor usage rate, and thus this solution is not economical. The last three columns of Table 5.3 show the critical path delays for benchmark circuits under three buffer distribution models. For the overflowed buffers, we remove them from buffer insertion solution, and then calculate the critical path delay. The delay model employed is the Elmore delay described in Section 2.4.1. We can find there are not much difference in the timing performance of the three buffer capacity models. Although there are some buffer overflow for the UNI model, but rip-up of those buffers does not affect the timing performance much. This is because (a) the number of overflowed buffers is small, (b) they may not lie on the critical path. These facts legitimize the use of the *uniform* buffer capacity model based on our estimation in structured ASICs; we can thus acquire an adequate and economic buffer solution with great regularity to improve the manufacturability and reduce cost, while not affecting the timing performance significantly.

Figure 5.5 further shows the two dimensional buffer distribution for a specific representative circuit, *pdc*. The three graphs show, respectively, the estimated buffer capacity distributions for nonuniform and uniform models, the actual buffer usage distribution for uniform model and the

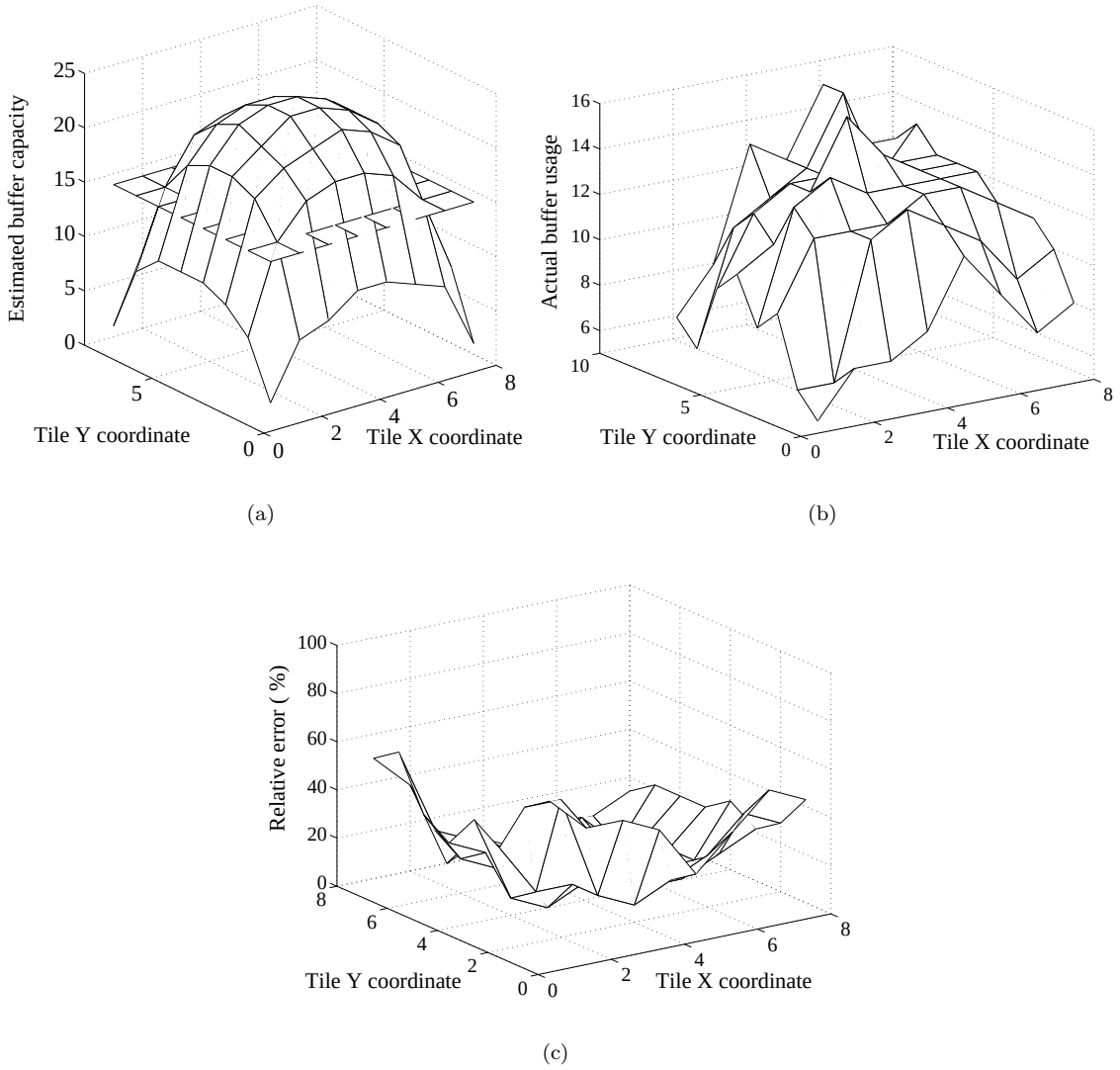


Figure 5.5: Accuracy of the statistical buffer distribution estimation. (a) Estimated buffer capacity distributions for the nonuniform and uniform distributions. (b) Actual buffer usage distribution under the uniform buffer capacity. (c) Relative error of the buffer capacity estimation compared with actual buffer usage under the uniform buffer capacity.

relative errors. We can see from Figures 5.5(a) that the distribution curve for the nonuniform distribution is of a "bell curve" form, but with a flat region in the center and a sharp dropoff near the boundary. Thus it does not deviate much from an average uniform distribution as shown in the figure. This property makes it reasonable to use a uniform buffer distribution based on the average of the nonuniform distribution, but still get good buffer usage and small buffer overflow. Figure 5.5(b) shows that the distribution of the actual buffer usage under the uniform buffer

distribution model. For this and all other benchmarks, the general trend is that the usage of buffers fits the uniform buffer capacity well in most part of circuits, but less buffers are used at the periphery. This is due to the fact there is generally less interconnect wires at periphery. In practice, we may preallocate some buffer areas on periphery to be decoupling capacitors to save some resources.

The relative error curve in Figure 5.5(c) shows that the estimated buffer capacity is generally a little higher than the actual buffer number, because our estimation is aimed at the maximum buffer capacity for a range of circuits, and naturally builds in some pessimism. Also this error is seen to a greater degree near the periphery, due to the smaller number of interconnect wires around there. However, for the most part, this error is less than 20%, which shows that our *a priori* buffer distribution estimation provides an economic solution.

5.5 Conclusion

In this chapter, we have proposed an early statistical buffer distribution estimation using Rent's rule, a simplified L/Z-shaped routing model, and a distance-based buffer insertion model. Based on this estimation, we presented a uniform buffer distribution method of great regularity to place prefabricated buffers on structured ASICs. Experimental results show that the buffer distribution estimation and method, although not based on physical design details, can accurately and economically plan buffer resources on structured ASIC chip, and it is shown the buffer capacity prediction matches actual buffer usage well.

Chapter 6

CMP-aware Routing for Minimized Dummy Metal Fills

Apart from performance challenges, modern VLSI interconnects design must take manufacturability issues into early design stages, so that a designed chip can be manufactured at a reasonable cost and with high yields under nanoscale fabrication technology. Oxide chemical-mechanical polishing (CMP) is a critical step that enables long-range oxide planarity in nanometer fabrication. The effectiveness of the planarization of the interlayer dielectric (ILD) through CMP is enhanced by appropriately inserting dummy metal fills in each metal layer. However, it is well-known that the inserted dummy metal can greatly affect interconnect performance, and can result in an explosion in the amount of data in the mask database. We propose a novel CMP-aware algorithm that explores opportunities encountered during routing, so that the routed metal layout requires fewer dummy metal fills to achieve oxide planarity. Our approach is embedded within a routing algorithm as a two-step solution. First in global routing, we apply a deflated routing capacity to distribute metal wires more uniformly; this is followed by a layer-assignment method optimizing a novel dummy fill metric which predicts the amount of dummy metal fills inserted for CMP process. Experimental results show that, with the same maximum allowable oxide height variation from CMP process, the amount of dummy metal required in the layout from our routing

algorithm can be up to 31.6% less than those from a regular router, showing that our routing algorithm produces a more manufacturable interconnect design.

6.1 Introduction

As VLSI fabrication technologies advance into the nanometer regime, chip design faces many new challenges, not only from deep-submicron physical effects such as timing, signal integrity and power problems, but also from new manufacturability issues, i.e., whether a designed chip can be manufactured at a reasonable cost and with high yields. Aggressive technology scaling has made it more difficult than ever before to resolve all manufacturability issues during post-layout processing. Therefore, for nanoscale circuits, it is important to incorporate manufacturability considerations in early design stages, using design for manufacturability (DFM) techniques [49].

Chemical-mechanical polishing (CMP) is a critical step that affects the manufacturability of a VLSI chip. This is a processing technology that enables long-range oxide planarization, and is applied at several steps during fabrication. The CMP process is especially crucial for interlevel dielectric (ILD) layers, as ILD variations must be carefully controlled due to aggressive lithographic depth-of-field focus budgets, and this variation can be further exacerbated by the multi-layer accumulation effect over successive dielectric layers [108]. At the same time, ILD thickness variations, and the associated dummy fills (to be detailed described shortly) used to reduce these variations, can also greatly impact circuit performance [57]. This chapter considers the problem of CMP-aware routing that has the ability to reduce the amount of metal fill inserted for ILD variation control. By incorporating CMP models in *both* global routing and layer assignment phases, our DFM-based routing approach is conducive to a more robust, manufacturable design.

To achieve post-CMP planarization, it is required that the pattern density prior to CMP process must satisfy some requirements. This necessitates the insertion of *dummy fills*, which typically have no electrical functionality, into spaces in the layout to generate a desired density distribution. In each metal layer, the dummy features inserted correspond to metal fills in the form of polygons, thus we use terms “dummy fill” and “metal fill” interchangeably in this chapter.

Dummy fills are usually inserted during the post-layout phase, and they can greatly enhance ILD planarity. However, they can create serious performance problems due to increased interconnect coupling capacitance, signal delay, crosstalk noise and power consumption [76, 105]. Moreover, such performance degradation in the post-layout stage can significantly harm design closure and result in costly respins. Further, the large number of dummy fills also creates difficulty in mask data processing and design verification.

The problem of optimal CMP fill insertion suffers from the classic tug-of-war between the level of available design detail and the flexibility available in changing the design for improved optimality. During late stages of the design cycle, a great deal of precise knowledge is available about the layout, but the flexibility for change is limited. The literature has focused on using dummy fill insertion as a post-processing stage, after global and detailed routing is complete; the works in [21, 73] optimize the geometric forms and locations of metal fills to improve the timing and crosstalk performance of circuit. In contrast, dummy fill optimizations that occur earlier in the design cycle must operate under incomplete layout information, but offer the potential for much more overall improvement. Unless these earlier steps in physical design are made CMP dummy fill aware, the solution space after conventional global and detailed routing can be very limited, and can lead to suboptimal solutions. With increased design complexities, it is very difficult, or even infeasible, to defer the handling of these manufacturability issues to the post-layout stage, when interconnect layout are fixed and only minor modifications can be made.

The contribution of this chapter is a novel routing approach to address the CMP dummy fill problem. Our algorithm handles such challenges early in the physical design phase by integrating these considerations within the router. Since routing is the major step that is responsible for determining the spatial distribution of a vast number metal wires between interlevel dielectric, our CMP-aware routing algorithm seeks to generate a solution that has a more CMP-friendly initial metal wire distribution, so that the amount of dummy fills required during the post-layout phase is significantly reduced over a conventional router. In our development of this idea, we first propose a novel dummy fill metric to estimate the total amount of dummy fills to be employed for CMP

process. Based on this metric, our routing algorithm goes through both global routing and layer assignment stages with *CMP awareness*, to arrange appropriate paths and layers for nets, so that the total amount of metal fills required during ILD CMP process is minimized.

Dummy fill insertion methods can generally fall into two categories: rule-based approaches, in which local pattern densities are enforced by design rules to be within some specific range, and model-based approaches, where the dummy feature density distribution is determined from analytical expressions as well as post-CMP oxide thickness constraints [20, 64, 108]. The rule-based approach has the disadvantage that the range for allowed density is fairly large, and the prescribed amount of dummy fills will usually go through trial-and-error for every design [108]. On the other hand, the model-based approach provides a direct and more accurate way to optimally allocate dummy fill, and is amenable to being used in optimization. Therefore, a model-based approach is employed in this chapter, both in developing a dummy fill metric that can be used during routing, and in evaluating the effectiveness of the results of our method.

Our approach goes through the global routing, layer assignment, and detailed routing steps: the CMP-awareness has been injected into the first two steps, and it is important for both of them to be CMP-conscious, otherwise any gains in one step may be lost in the other. Our results are finally validated by comparing the number of dummy fills that must be inserted for our solution, as against the conventional approach.

The rest of the chapter is organized as follows: The CMP-friendly routing problem is formulated in Section 6.2. Section 6.3 reviews model-based CMP dummy fill methods, and proposes a novel dummy fill metric to estimate the dummy fill amount. In Section 6.4, our CMP-friendly routing algorithm for minimized dummy fills is presented. Experimental results are discussed in Section 6.5, and Section 6.6 concludes the chapter.

6.2 Routing Consideration for Dummy Metal Fill

The routing phase begins with a placed netlist and connects the pins of the signal nets, whose locations have been determined by the placement, and determines a detailed route for each net. In

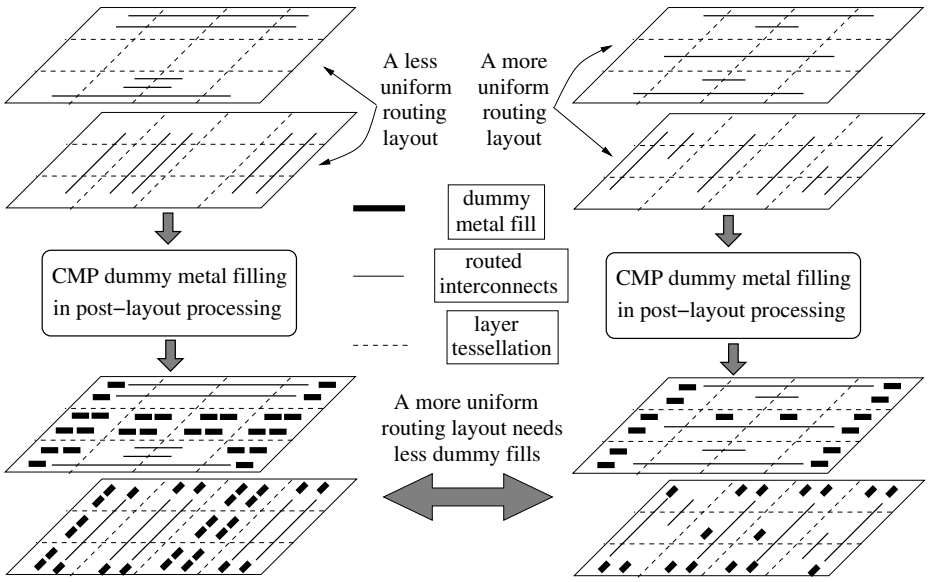


Figure 6.1: A more uniform initial metal distribution requires fewer CMP dummy fills during the post-layout process. The two layouts are two different routing schemes on a pair of routing layers for the same placement, but the layout on the left is less uniform than that on the right, and requires a larger amount of dummy fill.

other words, the precise routes and routing layers of all nets are assigned, and the distribution of the metal wires is determined after this step. Traditional objectives for routing have included wire length, timing, and routability. A routing result defines the initial metal density distribution[1], and once this is known, the dummy fill insertion procedure during the post-layout stage can proceed with precise input information, to determine the optimal dummy metal distribution for the given routing solution.

It is desirable to control the amount of dummy fill that is inserted, since this results in a lower interconnect performance penalty. However, as mentioned in Section 6.1, waiting until the routing step is completed severely limits the flexibility that is available in controlling the amount of dummy fill to be inserted. On the other hand, if routing is made CMP-aware, it can actively attempt to ensure that in addition to the traditional objectives, it also attempts to minimize the total dummy fills inserted.

Our work does precisely this, and addresses this optimization during routing. Our approach is based on the following observation, which can be formally proved as a simple consequence of

[1]Along with the signal nets, the power grid is also distributed on the same routing layers, but we can generally assume power grid is a *uniform* wiring network on each routing layer.

the theorems in Section 6.3.2: if a more uniform initial interconnect wire distribution is generated during routing, the total amount of dummy fills required during the post-routing fill process will tend to be lower; otherwise, a non-uniform initial metal distribution will likely require more dummy fills to meet the CMP manufacturability. This is illustrated by an example in Figure 6.1, where the less uniform layout on the left requires more dummy fills than the less uniform layout on the right.

Hence, uniformity becomes an additional important goal that must be emphasized during routing for better manufacturability of interconnect design. Routing distributes metal wires to various lateral locations on chip, as well into different metal layers, and thus a uniform metal wire distribution from the viewpoint of routing actually has two senses of meaning:

- xy-uniformity: Wires should be distributed more uniformly in lateral x and y directions on layout plane;
- z-uniformity: Wires should be distributed more uniformly in z direction, i.e., wires should be distributed evenly among different metal layers so that metal density distribution is more uniform in each layer.

Two caveats should be kept in mind with regard to the above notions of uniformity. First, enforcing uniformity is different from the congestion reduction objective. Congestion cost functions typically thresholded: they are zero until we come close to the congestion constraint, where the cost function increases and becomes very large for a constraint violation. This is clearly not the same as the uniformity requirement that is important for reducing dummy fills. Second, this uniformity should not be confused with the *post-fill* uniformity that that enforced by a rule-based fill algorithm. Our concept of uniformity is enforced at the *pre-fill* stage, and we use a model-based approach to develop the metric, introduced in Section 6.3.2. We will show, through an example, that two density distributions, which are similarly "uniform" to each other under common uniformity measures such as standard deviation, can show rather different uniformity behaviors under our metric, and have significantly different amounts of metal fill. Based on our metric, the uniformity is actually measurable and can be embedded into the optimization process to guide routing and

layer assignment.

Based on these observations, our CMP-aware routing problem can be formulated as follows. The input to the routing problem includes:

1. a given placement for the circuit netlist,
2. multiple metal layers for interconnect routing, and
3. processing technology parameters.

Furthermore, we assume that all routing layers have similar electrical properties[2]. Each layer is assumed to have a dedicated routing direction that is either horizontal or vertical; two neighboring routing layers have different dedicated directions, so as to form a routing *tier*. To minimize via usage for better circuit reliability, we further require that each net uses only one tier in detailed routing. We present a complete routing solution that includes global routing, layer assignment and detailed routing, and the routing solution is required to meet traditional requirements, such as routability and wire length, in addition to being DFM-aware in generating a *more uniform* distribution of wires, which minimizes the number of dummy fills inserted in post-layout processing.

6.3 CMP Dummy Metal Fill Model and Metric

6.3.1 Oxide CMP Model

The oxide topography after CMP can be computed using model-based simulations. Several models for oxide planarization with CMP are reviewed in [87], and the simple, yet accurate closed form expression reported in [89] is one of the most widely used. This model describes the relationship between post-CMP oxide thickness and pre-CMP feature density distribution, and bears the following form:

$$z = \begin{cases} z_0 - K_i t/\rho(x,y) & t < \rho z_1/K_i \\ z_0 - z_1 - K_i t + \rho(x,y) z_1 & t > \rho z_1/K_i \end{cases} \tag{6.1}$$

[2]The work can be extended to handle cases where the electrical properties of different layers are dissimilar.

where

z = post-CMP oxide thickness;

K_i = blanket oxide polishing rate;

z_0 = thickness of oxide deposition;

z_1 = initial step height;

t = total polish time;

$\rho(x, y)$ = initial effective metal pattern density before CMP.

Normally the total polish time is larger than $\rho z_1/K_1$, and thus the final oxide topography variation is determined by the initial effective metal density $\rho(x, y)$, which is defined as a weighted sum of local metal pattern densities within a neighborhood region. We can further discretize the problem by tessellating the chip area into an array of $m \times n$ small rectangles of equal dimensions: this is referred to as the CMP grid, and each such rectangle is denoted as a CMP cell. In the multiple ILD layer case, each layer is tessellated into a CMP grid of the same size. The local metal density in a CMP cell can be obtained by taking into account all the metal wire objects in the cell. Furthermore, the effective metal density in a CMP cell can be expressed as a convolution between $d(i, j)$, the total local metal density obtained in cell (i, j), and $f(i, j)$, the discrete form of the weighting function. In the frequency domain, the convolution is a product, and this can be expressed as:

$$\hat{\rho}(i, j) = \hat{d}(i, j) \cdot \hat{f}(i, j), i = 0, \ldots, m-1; j = 0, \ldots, n-1; \tag{6.2}$$

in which "$\hat{}$" is the Discrete Fourier Transform (DFT) operator. According to [108], we can empirically represent the weighting function on the CMP grid as a discretized elliptical function:

$$f(i, j) \approx c_0 \exp(c_1(i^2 + j^2)^{c_2}), i, j = -R, \ldots, R \tag{6.3}$$

in which c_0, c_1 and c_2 are constants specified for each fabrication process, and R characterizes the region size of the truncated elliptic curve.

Figure 6.2 shows a CMP grid of size $m \times n$, with an elliptic weighting function centered at

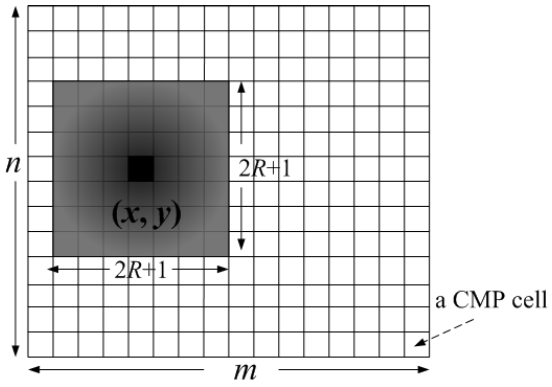


Figure 6.2: A CMP grid of size $m \times n$ and a discretized weighting function.

position (x, y); the shading in the figure indicates the strength of the weighting factors: we can see that this fades from the the center (x, y) to the periphery. The initial feature density distribution, obtained after routing, is modified by insertion of dummy metal fills, and we can further write equation (6.2) into a more specific convolution form[3]:

$$\rho(i,j) = (d^0(i,j) + d'(i,j)) \otimes f(i,j) \quad = \quad \sum_{i'=i-R}^{i+R} \sum_{j'=j-R}^{j+R} (d^0(i,j) + d'(i,j)) \cdot f(i-i', j-j') \\ i = 0, \ldots, m-1; j = 0, \ldots, n-1; \tag{6.4}$$

in which "$\otimes$" is the circular convolution operator, $d^0(i,j)$ and $d'(i,j)$ are the initial metal density and the dummy fill density for CMP cell (i,j), respectively, and m and n are the number of CMP grids in the the two orthogonal directions. The above equations can model ILD CMP effects under a single oxide layer scenario; however, with multiple ILD layers in an actual chip, the variation of ILD thickness shows an accumulation effect due to the multi-layer structure. In [119], it is modeled as an attenuated propagation of variation in one layer through the subsequent CMP processes, as:

$$\hat{\rho}_k = \begin{cases} [\hat{d}_k + (z_{k-1}/z_k)\hat{\rho}_{k-1}] \cdot \hat{f}, & k > 1 \\ \hat{d}_1 \cdot \hat{f}, & k = 1 \end{cases} \tag{6.5}$$

where ρ_k is the effective density distribution at layer k; z_k is the initial step height for each layer k;

[3]The index in the summation is actually a revolving index. When it reaches outside the layout, its neighborhood corresponds to the next die, since dies form a regular two-dimensional array on a wafer. Thus, the operation is a circular convolution.

d_k is the combined metal density distribution at layer k, which include contributions from initial metal features as well as metal fills.

Based on equations (6.1), (6.3), (6.5) and (6.5), the work in [108] formulates a linear programming (LP) problem to determine the optimized dummy metal distribution before CMP, so as to address the control of manufacturability during CMP process. For single layer case, it has the form:

$$\begin{aligned} \text{minimize } & \sum_{i,j} d'(i,j) \\ \text{subject to: } & 0 \le \rho^{low} \le \rho(i,j) \le \rho^{high} \le 1 \\ & \rho^{high} - \rho^{low} \le \epsilon \\ & 0 \le d'(i,j) \le d^a(i,j) \end{aligned} \tag{6.6}$$

in which ρ^{high} and ρ^{low} are two auxiliary variables, and $d^a(i,j)$ is the maximum dummy fill density allowed at cell (i,j). A variation range, ϵ, is allocated to bound the ILD thickness variation in final topography and guarantee the CMP manufacturability. The total amount of additional dummy fills is minimized so as to maximize polish rate and thus throughput in CMP. For the multiple layer case, the ranged-variation problem is similarly formulated as in [108]:

$$\begin{aligned} \text{minimize } & \sum_{i,j,k} d'_k(i,j) \\ \text{subject to: } & 0 \le \rho_k^{low} \le \rho_k(i,j) \le \rho_k^{high} \le \sum_{l=1}^{k} (z_l/z_k) \\ & \rho_k^{high} - \rho_k^{low} \le \epsilon_k \\ & 0 \le d'_k(i,j) \le d_k^a(i,j), k = 1, \ldots, L \end{aligned} \tag{6.7}$$

in which L is the total number of layers and z_k is the initial step height for each layer k. The other variables have the same meaning as in problem (6.6), except for an additional k subscript to indicate an association with a specific layer k. Additionally, there is a specific variation range ϵ_k allocated for each layer $k, k = 1, \ldots, L$, and the objective is altered to minimize the total

amount of dummy fills on all layers. The problems formulated in (6.6) and (6.7) can be solved with commercial linear solver [108], or Monte-Carlo methods [20], for improved efficiency.

6.3.2 A Novel Dummy Fill Metric

The above CMP dummy fill model is embedded in our routing algorithm to optimize dummy fills. If one were given a specific metal distribution, the LP problem (6.6) or (6.7) could be solved to obtain the amount of dummy fills associated with that distribution. However, solving the LP problem involves a high computation expense, and this is not amenable to being embedded in the optimization process of the routing stage. Hence, we propose a novel metric to estimate the total dummy metal amount *without* directly solving the LP problem, which can greatly simplify the computation complexity. The metric is based on the following theorems related to LP problems (6.6) and (6.7), and here we use the same notation as in Section 6.3.1.

Theorem 1. *For the single-layer LP problem (6.6), when $\epsilon = 0$, if the DFT, $\hat{f}$, of weighting function f has no zero coefficient, and the LP problem is feasible, then the solution $d'(i,j)$ has the following property:*

$$\sum_{i,j} d'(i,j) = \sum_{i,j} (d^0_{max} - d^0(i,j))$$

where $d^0_{max} = max\ (d^0(i,j))$, for $i = 0, \ldots, m-1;\ j = 0, \ldots, n-1$, which is the maximum initial metal density distribution on CMP grid.

Proof. Denote $d(i,j) = d^0(i,j) + d'(i,j)$ as the total metal density at cell (i,j). According to equation (6.2) and the fact $\hat{f}(i,j)$ has no zero coefficient, we have $\hat{d}(i,j) = \hat{\rho}(i,j)/\hat{f}(i,j), i = 0, \ldots, m-1; j = 0, \ldots, n-1$. With $\epsilon = 0$, the effective density $\rho(i,j)$ is uniform, such that $\hat{\rho}(i,j) = \delta(i,j)$, in which $\delta(i,j)$ is the two-dimensional delta function. Therefore, we have $\hat{d}(i,j) = \delta(i,j)/\hat{f}(0,0)$, so that the total metal density distribution $d(i,j)$ is a uniform distribution. As problem (6.6) is a minimization problem, the minimized amount of additional dummy fills to achieve total density uniformity is $\sum_{i,j} d'(i,j) = \sum_{i,j} (d^0_{max} - d^0(i,j))$. □

In practice, the variation range constraint ϵ is of a very small value, which is close to the ideal

case of 0, and $f(i,j)$ is an empirical weighting function of elliptic form, whose DFT is generally a bell-shaped curve and does not have any zero coefficients. Therefore, with Theorem 1, it is reasonable to employ:

$$M_{single} = \sum_{i,j} (d^0_{max} - d^0(i,j)) \tag{6.8}$$

as a metric to predict the amount of dummy metal $\sum_{i,j} d'(i,j)$ in single layer CMP case, which we denote as single layer metric. The validity of employing the theorem with non-ideal variation constraint is discussed in later examples. Furthermore, we can extend to the case of multi-layer oxide CMP problem, and have the following:

Theorem 2. *For the multi-layer LP problem (6.7), when $\epsilon_k = 0$, $k = 1, \ldots, L$, if the DFT, $\hat{f}$, of weighting function f has no zero coefficient, and the LP problem is feasible, then the solution has the property:*

$$\sum_{i,j,k} d'_k(i,j) = \sum_k \sum_{i,j} (d^0_{max,k} - d^0_k(i,j))$$

where $d^0_{max,k} = max\ (d^0_k(i,j))$, for $i = 0, \ldots, m-1; \quad j = 0, \ldots, n-1$.

Proof. We first prove that: with the conditions given above, $d_k(i,j) = d^0_k(i,j) + d'_k(i,j)$ is a uniform distribution for each layer $k = 1, \ldots, L$.

When $k = 1$, according to equation (6.5), we have $\hat{\rho_1} = \hat{d_1} \cdot \hat{f}$. As $\epsilon_1 = 0$, ρ_1 is uniform distribution. Following the same derivation as in the proof of Theorem 1, we can conclude that d_1 is uniform. For layer $k > 1$, we can express the relationship of equation (6.5) in convolution form:

$$\rho_k = [d_k + (z_{k-1}/z_k)\rho_{k-1}] \otimes f = d_k \otimes f + (z_{k-1}/z_k)\rho_{k-1} \otimes f$$

As $\epsilon_{k-1} = 0$, ρ_{k-1} is uniform, hence $\rho_{k-1} \otimes f$ is also uniform. Since $\epsilon_k = 0$, ρ_k is uniform. Therefore, from the above equation, $d_k \otimes f$ is uniform. Following the the same derivation as in the proof of Theorem 1, we can conclude that d_k is *uniform* for $k > 1$.

So we can conclude that for each $k = 1, \ldots, L$, $d_k(i,j)$ have the *same* value for all $i =$

$0, \ldots, m-1;\ j = 0, \ldots, n-1$. Considering the fact that $d_k(i,j) = d_k^0(i,j) + d_k'(i,j)$, and problem (6.7) targets at the minimization of $\sum_{i,j,k} d_k'(i,j)$, we can conclude that the minimized amount of total dummy fills over all layers is $\sum_k \sum_{i,j} (d_{max,k}^0 - d_k^0(i,j))$. □

Theorem 2 provides the grounds to employ

$$M_{multi} = \sum_k \sum_{i,j} (d_{max,k}^0 - d_k^0(i,j)) \tag{6.9}$$

as a metric to estimate the total amount of dummy metal fills for CMP. The simple form legitimizes its use as a goal for optimization in layer assignment algorithm.

As discussed in Section 6.2, this dummy fill metric (6.8) and (6.9) is essentially a measure for uniformity. As compared with other commonly-used metric for uniformity characterization, our metric has a theoretical basis in the CMP model, and is an excellent predictor of the total dummy fills required. Figure 6.3 illustrates how our dummy fill metric correlates with the actual amount of dummy fills and the comparison with other uniformity metrics under a single layer dummy filling case taken here as an example. Two commonly-used uniformity metrics that we compare this against are standard deviation (STD) and average absolute error (AAE), and AAE is defined as $1/(mn) \sum_{i,j} |d^0(i,j) - \bar{d^0}|$, where $\bar{d^0}$ is the mean of all $d^0(i,j)$ over the CMP grid.

We compare the effects of dummy fill insertion to two metal distributions on a 13×13 CMP grid, labeled as "I" and "II". The two initial distributions (prior to dummy fill insertion) are randomly generated, with the same total amount of metal densities, but different levels of uniformity. In terms of simple measures of uniformity such as the STD and AAE, the two layouts, shown in Figure 6.3(a), are initially identical. For better visualization, we have shown the density maps at different points on the z-axis of the same graph throughout this figure. Equation (6.8) is employed to calculate the dummy fill metric for each distribution, and the respective values are shown next to Figure 6.3(a). For comparison, the STD and AAE values for each distribution are also listed. Visually, the two distributions are seemingly of similar uniformity, which is also confirmed by the little difference between their STD and AAE values. However, they show significantly different

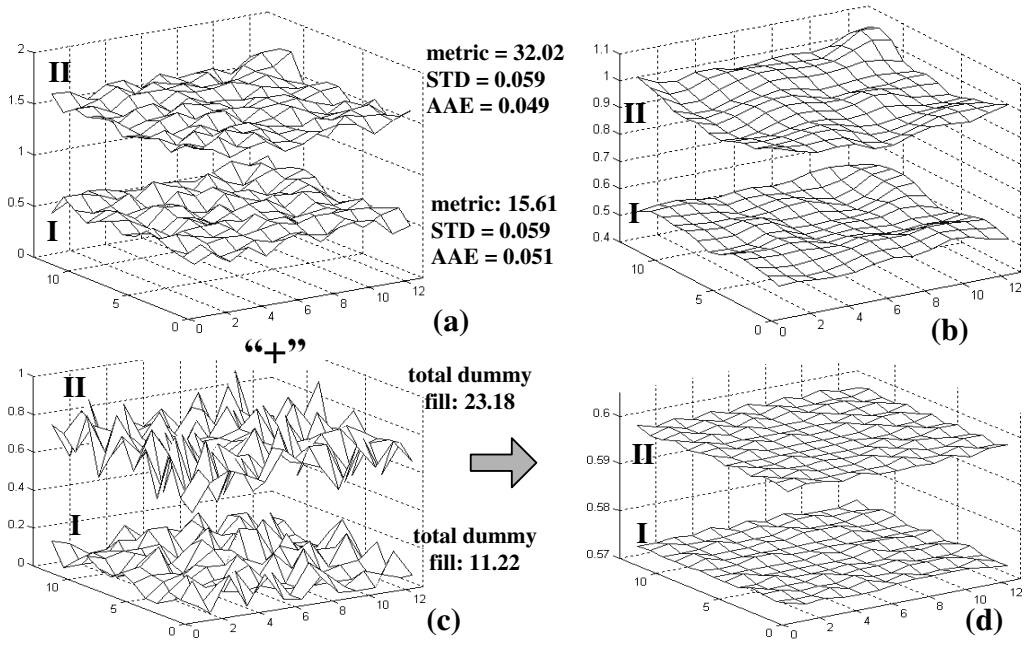


Figure 6.3: Uniformity metric, dummy fill metric and actual amount of dummy fills for two metal distributions. (a) Initial metal density distributions. (b) Initial effective density distributions. (c) Dummy metal fill distributions. (d) Effective density distributions after dummy fill insertion.

values under our metric, and this predicts they require vastly different amount of dummy fills during post-layout processing. Figure 6.3(b) shows the effective density distributions for each of these layouts, based on the convolution between the density function and the weighting function. Next, we solve problem (6.6), with $\epsilon = 10^{-3}$, to obtain the dummy fill distribution as well as the total amount of fill (the unit is the number of CMP cells). We find that the initial distribution II, which is less uniform under our metric, requires a larger amount of dummy fills than distribution I, and the actual amount of total dummy fills in both case correlate well with the predictions from our metric. Figure 6.3(c) and (d) plot the distribution of dummy fills and the effective metal density after dummy fill insertion, respectively; clearly, dummy fills significantly reduce the variation in the effective density distribution, and bring it in line with the ϵ specification. Thus, we see that our metric for uniformity is a good predictor of the amount of CMP fill required after layout, and that it is better than the other candidate uniformity metrics.

Based on this example, we also explore the insights for the validity of employing Theorem 1

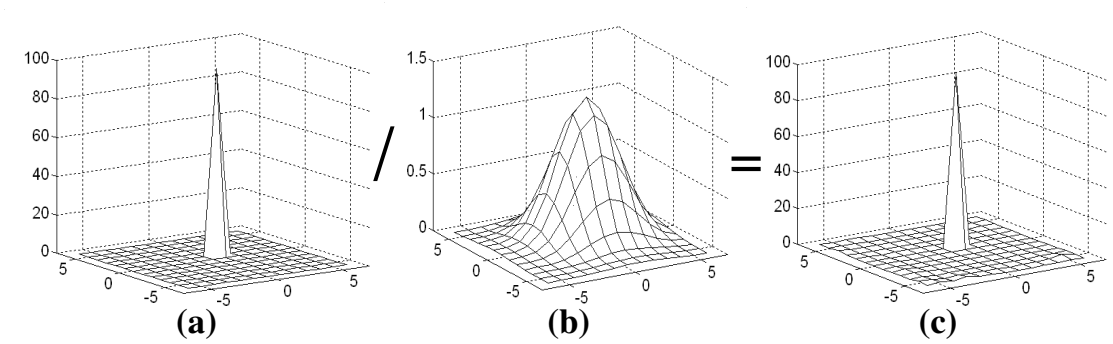


Figure 6.4: Plots of DFT coefficients for distributions in case I of the example in Figure 6.3. (a) $\hat{\rho}$: DFT of effective density ρ after dummy filling. (b) $\hat{f}$: DFT of weighting function f. (c) $\hat{d}$: DFT of total density d after dummy filling.

to predict the dummy fill amount even for the variation range constraint ϵ is not exactly zero. Figure 6.4 plots the curves for the DFT of effective density ($\hat{\rho}$) and the DFT of total density ($\hat{d}$) after dummy filling, as well as the DFT of weighting function ($\hat{f}$). The values of $\hat{\rho}$ and $\hat{d}$ are computed from the distributions in case I of the above example (we obtain similar results for case II). As shown in Figure 6.4(a), although the variation range constraint $\epsilon > 0$, the DFT of ρ still shows great resemblance to the two-dimensional delta function since ϵ is small, and obviously has a much narrower bandwidth than the weighting function f (shown in Figure 6.4(b)). Hence following the proof of Theorem 1, $\hat{d}(i,j) = \hat{\rho}(i,j)/\hat{f}(i,j)$, we can still safely assert $\hat{d}$ is very close to a delta function, which is verified in Figure 6.4(c). Moreover, although $\hat{f}$ has no zero coefficients, it reduces to near-zero after a certain distance; however, since $\hat{\rho}$ goes to zero faster, and hence our argument in the proof of Theorem 1 holds good even in the presence of limited numerical accuracy. Therefore, the total density distribution d is still approximately a uniform distribution, thus we can safely employ Theorem 1 to predict the dummy fill amount when variation constraint $\epsilon > 0$, but of very small value.

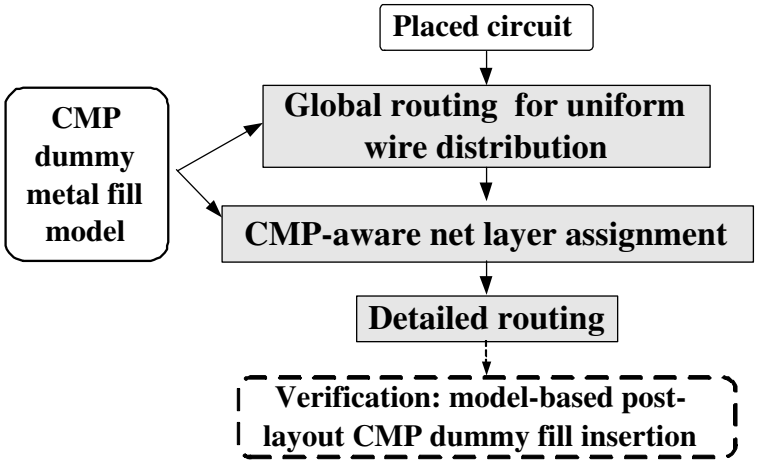


Figure 6.5: Overall flow for the CMP-aware routing algorithm and CMP metal fill verification process.

6.4 Routing Algorithm for Minimized Amount of Dummy Metal Fill

6.4.1 Routing Model and Algorithm Overview

Our routing algorithm is a complete routing flow, which includes three phases, as shown in Figure 6.5: global routing, layer assignment, and detailed routing. In the first global routing phase, we build the global routing graph as in Section 2.2.1. During the process, the chip is tessellated into a two-dimensional array of GRCs, or a routing grid. In general, this grid may be different from the CMP grid discussed earlier; however, we will show that there are some conveniences associated with using the same grid for both purposes. The global routing phase determines the rough routing regions, in terms of GRCs, that each net passes through, while satisfying the routing capacity requirements. At the same time, with the CMP dummy fill model that feeds into global routing, nets are also spread out laterally using a routing capacity deflation (RCD) technique to obtain a better xy-uniformity of metal distribution for minimized dummy fills.

Global routing is followed by the layer assignment phase, during which each net is assigned to an appropriate tier. By optimizing the dummy fill metric (6.9), layer assignment distributes nets evenly among layers so that z-uniformity is achieved. During this process, it is ensured that

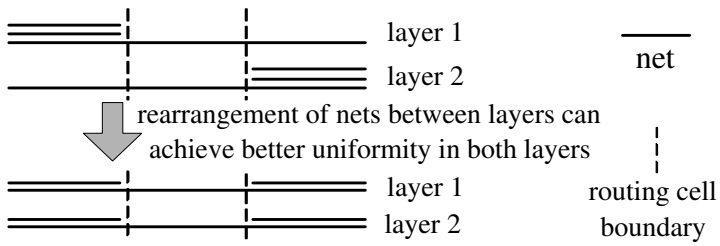


Figure 6.6: An illustration that the uniformity on both layers in a two-layer example can be improved by rearranging the nets between layers.

routability constraints are also met. As an integrated flow, The final phase performs detailed routing for each net in the tier it was assigned to, and determines a detailed path for it as well as resolves routing overflow. After the routing process, we perform CMP dummy fill insertion, for which we employ the model-based dummy metal filling method proposed in [108], and solve problem (6.7) to obtain a dummy fill distribution.

Since layer assignment plays an important role in distributing metal wires to appropriate layers, and can achieve significant reduction in CMP dummy fills, it is a core part of our CMP-aware routing algorithm. Hence we will first describe this phase in this chapter; the global routing and detailed routing phases are discussed following that.

6.4.2 Layer Assignment for Minimized Dummy Fills

Global routing determines the rough route for each net in terms of GRCs. However, before the detailed route is found, we must perform layer assignment to determine the tier (defined in Section 6.2) on which each net will be detailed-routed. In fact, layer assignment not only assigns a tier for each net to guarantee routability in the detailed routing phase, but also plays a very important role in achieving metal distribution uniformity in each layer, i.e., the z-uniformity, so as to reduce the amount of CMP dummy fills. To show the effectiveness of layer assignment in achieving wiring uniformity, Figure 6.6 shows a toy example for layer assignment in one-dimensional routing with two routing layers. In the example, the routing capacity of each layer is three, which is not violated in either case; however, by wisely rearranging nets between two layers, we can achieve a more uniform and CMP-friendly metal distribution on both layers.

To simplify the process of metal utilization planning in layer assignment, we tessellate the CMP grid so that it is of the same dimension as the global routing grid[4] such that each CMP cell corresponds to a GRC of the same size. As global routing finds a routing path for each net in terms of GRCs, or CMP cells, we can estimate the metal density contribution to a CMP cell from globally routed nets. Therefore, once a layer assignment for nets is known, the metal density distribution for each metal layer can be obtained. Based on this distribution, our layer assignment can quantitatively evaluate and measure z-uniformity using the dummy fill metric (6.9), and can minimize this metric during this process. As layer assignment deals with multiple layers, in the following discussion, we map the two-dimensional global routing grid defined in Section 2.2.1 to each metal layer, and update the routing edge capacity at each layer to count only the capacity contribution from that layer. Recall that we group pairs of consecutive layers, with orthogonal routing directions, into a tier. Under our CMP-aware routing scenario, the layer assignment problem can be formulated as the following partitioning problem:

Given a set of p nets $N = \{n_0, \ldots, n_{p-1}\}$ that have been globally routed, the layer assignment problem partitions N into q disjoint subsets $N_0, \ldots, N_{q-1}$, which correspond to q tiers in circuit, so as to minimize the dummy fill metric (6.9), while ensuring that the nets in each subset (tier) should meet the routing capacity constraint on each tier.

Our procedure here assumes that each tier is electrically identical. In case this is not the case, an additional penalty component can be added to the cost function to discourage critical or long nets from being routed on layers with higher delay characteristics.

The Layer Assignment Algorithm

Next, our layer assignment problem seeks to partition the set of all globally routed nets to q disjoint subsets, which correspond to q tiers in design. With each net assigned to a unique tier, the layer assignment algorithm will minimize the dummy metal fill metric (6.9) while meeting the routing capacity constraint for each layer. We can show the NP-hardness of this problem by

[4]These two grids are actually of very similar geometric coarseness from the computational experience in [108], which justifies our usage of same dimension for both global routing grid and CMP grid.

Algorithm: `Layer_assignment_for_minimized_dummy_fill_metric`

Input: A set of p globally routed nets and q tiers.
Output: Each routed net is assigned to a unique tier, so that dummy fill metric (6.9) is minimized, and routing capacity constraint is met for all layers.

```
begin
1.    Randomly assign each net to one of q tiers, with equal probability;
2.    Check routing capacity violation for each layer;
3.    for (each violated routing cell boundary) {
4.      do
5.        Pick the shortest untried net over the boundary and try to move to subsequent tiers to fit in
          a new suitable tier;
6.      until (routing violation fixed or all nets over the boundary have been tried)
7.    }

8.    while(true) {
9.      Unlock all nets;
10.     has_one_move = false;
11.     while(true) {
12.       best_move_gain = +∞; best_net = NULL; best_tier = NULL;
13.       for (each unlocked net i) {
14.         for (each tier t_o other than currently assigned tier t_i) {
15.           if (congestion violation < δ when move net i from tier t_i to tier t_o) {
16.             move_gain = get_move_gain(i, t_i, t_o);
17.             if (move_gain < best_move_gain)
18.               best_move_gain = move_gain; best_net = n; best_tier = t_o;
19.           }
20.         }
21.       }

22.       if(best_move_gain < −λ) {
23.         has_one_move = true; Lock best_net;
24.         move best_net to best_tier; update metal density distribution;
25.       }
26.       else
27.         break;
28.     }
29.
30.     if (! has_one_move)
31.       break;
32.   }
end
```

Figure 6.7: Layer assignment algorithm for minimized dummy fill metric.

reducing from the bin packing problem [45]. Therefore, we propose a greedy-based heuristic, whose pseudocode is outlined in Figure 6.7.

Based on the idea that a "uniform" distribution will result in better manufacturability, the heuristic first randomly distributes each net to one of the q tiers, with equal probability, to generate an initial layer assignment solution. Line 3 through 7 in the algorithm then fixes routing capacity violations by moving the shortest untried net passing through a violated grid cell boundary, to another tier, and keeps trying this until the violation is fixed; or if overflow can not be resolved, we will leave to it be fixed by rip-up-and-reroute in detailed routing. Based on this initial solution, a

Algorithm: `get_horizontal_move_gain`

Input: The density map $Map_{h,i}$ and total density $D_{h,i}$ for horizontal part of net i; current layer number c and new layer number d; max density $d^0_{max,c}$, $d^0_{max,d}$ and their position P_c, P_d for current and new layer, respectively; and their position P_c, P_d for current and new layer, respectively; CMP grid size $m \times n$;
Output: Dummy fill metric gain for moving horizontal part of net i from layer c to d.

```
begin
    /* Calculate metric gain for addition to new layer*/
1.  Traversing only Map_{h,i} region, update and find the difference Δ_d between new max density
    and d^0_{max,d} for layer d by addition of net segment;
2.  move_gain = m × n × Δ_d − D_{h,i};
    /* Calculate metric gain for removing from current layer*/
3.  if (P_c ∈ Map_{h,i}) {
4.      traversing whole CMP grid region, update and find the difference Δ_c between new max density
        and d^0_{max,c} for layer c by removal of net segment;
5.      move_gain += (m × n × Δ_c + D_{h,i});
6.  }
7.  else
8.      move_gain += D_{h,i};
9.  return move_gain;
end
```

Figure 6.8: Algorithm to calculate dummy fill metric gain by moving the horizontal part of a net to another layer.

greedy-based heuristic is then performed to move nets among tiers so as to minimize the dummy fill metric (6.9). There are two nested loops in the greedy heuristic. For the inner loop from line 11 to 28, each time, a net with most negative metric gain is picked and moved to new tier, and we lock this net so that it will be fixed and not considered later. Even if this move results in a small routing capacity violation in the new tier, as long as it is less than a small number δ, we still allow it, in the hope that this will encourage the reduction of dummy fill metric, and that a later detailed routing step can fix this small violation. The operations in the inner loop greedily move nets until a cut-off point is reached, where best gain is trivial (when gain $< -\lambda$, where $\lambda > 0$ is a pre-specified small number). Empirically, we have observed the best gain no longer shows any significant improvement once the inner "while" loop iterates beyond the cut-off point. The outer loop from line 8 to line 32 resets the lock status of all nets to "unlock" for another run of the inner loop, and it stops unless there is a nontrivial metric gain in the inner loop. In practice, we have observed that the outer loop always stops after less that ten iterations.

The "get_move_gain" function of line 16 in Figure 6.7 returns the metric gain if a net is moved to a new tier, and is called in the inner loop of the heuristic. Hence, we design an efficient implementation based on the structure of metric (6.9). Moving a net to another tier involves

displacement of both horizontal and vertical parts of the net, and they belong to neighboring routing layers. For conciseness, Figure 6.8 only shows the calculation of metric gain due to the moving of horizontal part; the vertical case follows similar steps. As input to the algorithm, we assume the metal density distribution for all layers gets updated with each actual net move; further, we assume the density map of each net i is calculated a priori: we refer to the density map for the horizontal segment of net i as $Map_{h,i}$. Here the density map for a net segment is a collection of all the CMP cells that have a non-zero metal density contribution from the net segment.

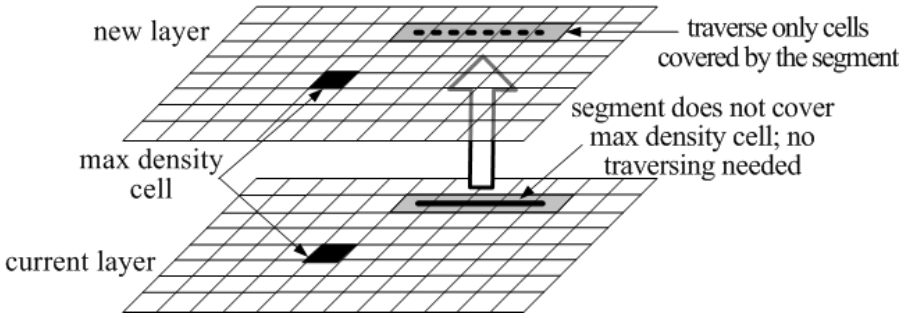


Figure 6.9: Illustration of metric gain calculation while moving a horizontal net segment to a new layer. Dark region corresponds to the density map for the net segment.

From equation (6.9), the dummy fill metric for a metal layer l bears the following form: $\sum_{i,j}(d^0_{max,l} - d^0_l(i,j))$, where $d^0_l(i,j)$ is the metal density at CMP cell (i,j) on layer l, and $d^0_{max,l}$ is the maximum value among them. The algorithm in Figure 6.8 makes use of the special structure of this expression to simplify computational complexity. The metric gain is computed separately for adding (lines 1-2) and removing (lines 3-8) stages. At line 1, in adding the horizontal segment of net i to a new layer d, we only need traverse the density map $Map_{h,i}$ to identify new maximum density, instead of traversing the entire CMP grid, because the new maximum density can be only at two locations: either it coincides with the old one, or it emerges at a position in the density map $Map_{h,i}$. This is illustrated in the upper part of Figure 6.9. With new maximum density known, line 2 computes the metric gain with a simple algebraic calculation. While removing the same part from a layer, we have to traverse the whole grid as in line 3-5 to update the new maximum density when current maximum density spot is covered by the density map $Map_{h,i}$, because under such scenario, the new max density can be anywhere in the grid due to the removal of the segment. For

other cases, it can be seen that the maximum density will remain same, hence it merely take an $O(1)$ complexity computation to obtain metric gain as line 8 suggests, which is also depicted in the lower part of Figure 6.9. Therefore, our algorithm for calculating the metric gain from a net move can significantly reduce traversal computations compared to enumerating the whole CMP grid for all cases and greatly improve efficiency.

6.4.3 Global Routing for Uniform Metal Distribution

Global Routing Flow

Our global routing seeks a feasible path on the routing graph for each net to satisfy the requirement such as congestion and wire length etc. As an important step to distribute metal wires on layout plane, our global routing also focuses on arrange nets in a more laterally uniform way so as to achieve xy-uniformity and utilize less dummy metal fills during ILD CMP process. The global routing flow first builds a minimum spanning tree (MST) based on Manhattan distance between pins, using Prim's algorithm [35], for each multiple-pin net, so as to decompose it into a set of two-pin nets (for simplicity, two-pin net will be simply referred to as net in the following phases). The order of routing those nets gives priority to shorter nets, i.e., by ranking all nets by their bounding-box wire length, our global router processes nets with ascending order of wire length. As observed in [67], longer nets have more flexibility to avoid congested regions, and therefore this scheme can reduce congestion as well as total wire length. If there is still an overflow at the end, we perform rip-up and reroute on all nets with the same order, as in [86].

Routing Capacity Deflation for Uniform Metal Distribution

Each two-pin-decomposed net is routed using Dijkstra's algorithm [35] to identify the minimum cost path connecting the two pins. As discussed in Section 6.2, a uniformly distributed interconnect structure is conducive to fewer dummy metal fills, which leads to a more manufacturable design. The cost function for the global routing algorithm must not only achieve routability, i.e., avoid overflow on routing edges, but also should create a better xy-uniformity to avoid gener-

ating high wire density regions. The cost function $Cost_e$ that we use for passing an edge e in the routing graph is defined as:

$$Cost_e = C_{length}(e) + \alpha \cdot C_{cong}(e) + \beta \cdot C_{bend}(e) \tag{6.10}$$

in which $C_{length}(e)$ is the length cost, $C_{cong}(e)$ is the congestion cost, and $C_{bend}(e) \in 0, 1$ is the cost for introducing a bend, and its value depends on whether there is a bend in the propagation direction; this is employed to suppress the number of vias for better interconnect reliability. The parameters α and β are user-defined, and generally we use large value for α to emphasize congestion avoidance and uniformity in wire distribution. The congestion cost $C_{cong}(e)$ takes the form: $C_{cong}(e) = \gamma^{(W_e - C_{rcd,\ e})}$, in which $\gamma > 1$ is a constant, W_e is the actual number of wires passing the boundary, and the function punishes overuse of routing resources on an edge. It is noted that, instead of using the actual edge capacity C_e, we utilize a *routing capacity deflation* (RCD) technique and intentionally deflate routing capacity to a value $C_{rcd,\ e} < C_e$. Plugged into the congestion cost, this reduced capacity can guide routing to target $C_{rcd,\ e}$ as the routing usage goal, hence encourage nets to spread over the routing plane laterally. In practice, the horizontal[vertical] deflated capacity is calculated by averaging the total estimated horizontal[vertical] wire length over all horizontal[vertical] edges. Thus defined deflated capacity can greatly help achieve a uniform wire distribution as the routing usage goal itself is calculated as an effort of "averaging".

6.4.4 Detailed Routing and Dummy Fill Verification

Global routing defines the rough routing region in terms of GRCs for each net, and the following layer assignment stage determines its specific routing tier; the obtained information is then fed to a detailed router. As the last step in an integrated routing flow, our detailed router finds the detailed routing paths for all nets and resolves the leftover congestion violations from previous stages. It employs a maze search algorithm, similar to the counterpart in [80]. Nets are detailed-routed following the same ascending length order as global routing, and the maze search algorithm finds the shortest feasible path composed of detailed routing tracks in the predefined

routing region and tier. Those nets that fail detailed routing will be ripped up, and rerouted in a "bloated" region to find a routable solution.

The routing solution defines a metal distribution in the layout. To verify the effectiveness of our efforts, we perform a CMP dummy fill insertion by solving the ranged-variation problem (6.7) to obtain the locations and number of dummy fills that are required to facilitate CMP. Our results, showing the outcomes of these experiments, are reported in Section 6.5.

6.5 Experimental Results

Our CMP-aware routing algorithm is implemented in C++, and the final dummy fill verification step employs CPLEX as the LP solver. The algorithm is tested on a Pentium 3.2GHz workstation with 2GB memory, on 6 benchmarks obtained from [34]. Table 6.1 lists some basic information about benchmarks and the routing grid (CMP grid) size.

Circuit	# pins	# nets	Grid	# layers
s5378	4734	1694	32 x 32	4
s9234	4185	1486	32 x 32	4
s13207	10562	3781	32 x 32	4
s15850	12566	4472	32 x 32	4
s38417	32210	11309	52 x 52	4
s38584	42589	14754	52 x 52	4

Table 6.1: Benchmark circuit parameters.

Our routing utilizes a model of four metal layers, and we use similar design rules in detailed routing as [80], except for some minor modifications to wire/via width and separation to accommodate the four-layer routing scenario. The CMP-related processing parameters are taken from [108]. Specifically, the step heights for all four layers are $z_1 = z_2 = z_3 = z_4 = 700$nm, and each layer k has a target maximum variation of $\epsilon_k = 30\text{nm}/z_k$ for $k = 1, 2, 3, 4$.

We compare the performance of our CMP-aware routing algorithm, which we refer to as *CMPR*, with two other approaches. The first comparison approach is taken from [80], which implements a multi-level routing system, and performs both global routing as well as detailed routing. However, it does not employ routing capacity deflation in global routing for more uniform wire distribution, nor does it utilize any layer assignment step to achieve better z-uniformity, and

we denote this approach as *MR* in data analysis. The second comparison algorithm is similar to the first one, except that a deflated routing capacity is employed in routing, and the deflated capacity is obtained based on the same average wire length estimation technique as in our algorithm. Since the routing in [80] is also congestion-driven, thus defined deflated routing capacity can help guide the routing towards a more uniform design, and achieve better lateral xy-uniformity; we denote this approach as *MR-mod*.

The same set of benchmarks are input to all three routing approaches, and the detailed routing results are then extracted to obtain metal density distribution for each metal layer. Based on those metal density distribution, we employ the CPLEX LP solver to solve the ranged-variation problem (6.7) and obtain the total amount of dummy metal fills.

Table 6.2 lists experimental results on total dummy fills and routing performance for all three algorithms. The first column lists circuit name, which is followed by the next set of three columns reporting the total amount of dummy metal fills necessary to meet the requirements in CMP fabrication process. The unit of the total amount of dummy metal fills is the number of CMP cells. As can be seen from the data, our CMP-aware routing (*CMPR*) algorithm can successfully reduce the total dummy fills necessary in the CMP process. By employing uniform global routing and layer assignment techniques, our *CMPR* algorithm can generate a routing solution requiring 15% - 30% less dummy fill than the *MR* approach, and these savings can be as much as 31.6% (*s38417*). The *MR* approach, without taking CMP issues into consideration during routing, consistently shows the largest amount of CMP dummy fills for all benchmarks. The *MR-mod* approach employs routing capacity deflation technique to generate a more uniform routing, and thus its routing results require less dummy fill than the *MR* approach. However, without addressing the z-uniformity, its routing solutions still require much more dummy fills than those from our *CMPR* algorithm.

The next set of three columns lists the routing completion rate, and all three approaches shows comparable completion rate which is 100% or very close. This is followed by a comparison of the total wire lengths for the three approaches and the number of vias used in detailed routing

Circuit	Total amount of dummy metal fill			Routing completion rate			Wire length ($\times 10^5$)			Number of vias			Run time (seconds)		
	MR	*MR-mod*	*CMPR*	*MR*	*MR-mod*	*CMPR*	*MR*	*MR-mod*	*CMPR*	*MR*	*MR-mod*	*CMPR*	*MR*	*MR-mod*	*CMPR*
s5378	496.1	452.1	409.7	100%	100%	100%	8.6	8.7	8.6	7043	7726	7436	1.07	1.46	3.24
s9234	372.0	340.0	309.9	99.71%	99.71%	99.68%	7.0	7.0	7.0	6046	6417	6091	0.76	2.21	2.76
s13207	400.6	384.4	334.4	99.91%	99.91%	99.91%	1.9	1.9	1.9	14904	16450	15118	2.61	3.84	13.79
s15850	332.6	291.5	240.1	99.93%	99.93%	99.92%	2.7	2.7	2.7	17878	20615	19959	4.50	6.26	12.31
s38417	925.6	807.3	633.1	100%	99.99%	100%	54.2	54.4	56.5	44894	45035	46119	12.66	13.37	30.70
s38584	624.6	612.3	528.6	99.99%	100%	99.89%	70.0	70.3	71.1	59670	59986	63845	37.45	39.55	111.62

Table 6.2: Comparison of dummy metal fill and routing performance results among our CMP-aware routing algorithm and two other comparison algorithms. *MR* = multilevel router [80], *MR-mod* = multilevel router with routing capacity deflation, *CMPR* = our CMP-aware routing algorithm.

in the next two sets of three columns. Both *MR-mod* and *CMPR* algorithms show slightly more via usage, and for some cases, longer total wire length, which is due to some routing detours that resulted from intentional deflation of routing capacity in global routing; however, this increase is not significant. Finally, the last set of three columns reports run time. Our *CMPR* algorithm requires a longer run time, which is attributable to layer assignment heuristic. However, with our efficient metric gain calculation method, the run time is limited to reasonable amount, and it is seen to scale almost linearly to the number of nets in problem.

To summarize Table 6.2, our *CMPR* routing algorithm can successfully generate manufacturing-friendly routing solutions involving much less dummy metal fills for CMP process. At the same time, it achieves similar wire length and via usage compared with other routing algorithms, and has reasonable run times.

6.6 Conclusion

In this chapter, we have proposed a novel routing for manufacturability problem which takes CMP process issues into consideration during routing. Our algorithm employs uniform global routing and layer assignment techniques to generate a more uniform metal density distribution, and together, we proposes a predictive metric for estimation of the amount of dummy metal fills. Experimental results show our CMP-aware routing algorithm can achieve up to 31.6% reduction of dummy metal fills for CMP process.

Chapter 7

Conclusion

This thesis has developed a series of routing and routing prediction algorithms to deal with various performance and manufacturability issues in building VLSI interconnects. We have developed a simultaneous shield and buffer insertion algorithm to reduce crosstalk noise during global routing, and have verified the fidelity of Devgan's noise metric so as to validly employ it. For a new family of 3D ICs, we have devised a temperature-aware global routing framework to effectively reduce peak temperature while achieving routability. Another contribution of this thesis is in proposing an accurate statistical buffer distribution estimator, based on which buffers are preplaced on structured ASIC templates, so that the number of buffers can economically satisfy buffering needs during the implementation phase of structure ASICs. A routing flow for optimized amount of post-layout CMP dummy fills has also been presented.

We have incorporated simultaneous shield and buffer insertion into global routing to mitigate crosstalk noise under a broad power/supply network paradigm. This method utilizes power wires as shields to reduce capacitive coupling, and employs buffers to block noise propagation and achieve better timing. To meet constraints imposed by limited routing and buffering resources, an iterative routing process is employed and during which, shield assignment and buffer insertion are considered simultaneously via a dynamic programming-like approach. We also demonstrate, Devgan's noise metric shows good fidelity and develop a noise margin inflation technique so that the metric can

be validly employed in our algorithm. Our algorithm achieves noise reduction improvements of up to 53% and 28%, respectively, compared to methods considering only buffer insertion, or only shield insertion after buffer planning.

A novel temperature-aware routing framework has been developed for 3D ICs to relieve the associated thermal problem. Our scheme makes judicious use of thermal vias and thermal wires to build good heat dissipation path to reduce temperature at hot spots while satisfying the requirement of routability. By employing adjoint sensitivity analysis, we have developed a technique to identify good thermal via/wire insertion locations. We have further applied a linear programming (LP) solver to best utilize thermal via/wire insertions at such locations to reduce temperature towards our desired goal while observing routability. The contention between routing and heat dissipation resources is resolved by performing the above analysis/insertion process and rip-up-and-reroute iteratively. Experimental results show that this global routing framework with thermal via and thermal wire insertion can effectively reduce the peak temperature in 3D ICs to meet the temperature requirement as well as alleviate routing congestion.

We have developed a statistical routing prediction technique and applied it to solve the buffering problem in structured ASICs. We have proposed a buffer insertion scheme so that prefabricated buffers can be inserted to a desired interconnect during implementation phase. A statistical buffer distribution estimator is then developed by utilizing Rent's rule, simplified L/Z-shaped routing model and length-based buffer insertion model. Based on this estimator, we have proposed a method to accurately and economically plan prefabricated buffers on structured ASIC chip, so that they can satisfy the buffering needs of global interconnects. This buffer distribution estimator has been verified on benchmarks and has been shown to be accurate and economical, and it matches the actual buffer usage well.

Finally, a new routing flow incorporating manufacturability considerations has been developed. By targeting uniformity during the routing process, this routing flow can generate routing solutions that require optimized amount of CMP dummy fills. Based on a CMP dummy fill model, we first propose a novel metric to quickly predict the amount of dummy fills and measure unifor-

mity. The CMP-awareness has been injected into the global routing and layer assignment stages. The global routing stage employs a routing capacity deflation (RCD) heuristic to achieve better lateral uniformity, while the layer assignment stage utilizes a greedy heuristic to distribute nets to different routing layers in a more uniform way and optimize the dummy fill metric. Finally, a detailed routing stage completes the routing flow and fix any leftover overflow. Experimental results show that, with the same maximum allowable oxide height variation from CMP process, the amount of dummy fills required in the layout from our routing algorithm can be up to 31.6% less than those from a regular router, showing that our routing flow produces a more manufacturable interconnect design.

Bibliography

[1] C. Ababei, Y. Feng, B. Goplen, H. Morgal, T. Zhang, K. Bazargan, and S. S. Sapatnekar, "Placement and Routing in 3D Integrated Circuits," *IEEE Design & Test of Computers,* 22(6), pages 520-531, November 2005.

[2] C. Ababei, H. Morgal, and K. Bazargan, "Three-Dimensional Place and Route for FPGAs," *Proceedings of the IEEE Asia and South Pacific Design Automation Conference*, pages 773-778, 2005.

[3] C. J. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 588-593, 1997.

[4] C. J. Alpert, A. Devgan, and S. T. Quay, "Buffer Insertion for Noise and Delay Optimization," *IEEE Transactions on Computer-Aided Design,* 18(11), pages 1633-1645, November 1999.

[5] C. J. Alpert, J. Hu, S. S. Sapatnekar, and P. Villarrubia, "A Practical Methodology for Early Buffer and Wire Resource Allocation," *Proceedings of the ACM/IEEE Design Automation Conference*, pages 189-194, 2001.

[6] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng, and D. Karger, "Prim-Dijkstra Tradeoffs for Improved Performance-Driven Routing Tree Design," *IEEE Transactions on Computer-Aided Design,* 14(7), pages 890-896, July 1995.

[7] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors,* 2003. (Available at http://www.itrs.net/reports.html)

[8] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors,* 2005. (Available at http://www.itrs.net/reports.html)

[9] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI,* Addison-Wesley, Reading, MA, 1990.

[10] K. Banerjee, M. Pedram, and A. Ajami, "Analysis and Optimization of Thermal Issues in High-Performance VLSI," *Proceedings of the ACM International Symposium on Physical Design*, pages 230-237, 2001.

[11] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat, "3-D ICs: A Novel Chip Design for Improving Deep Submicrometer Interconnect Performance and System-on-Chip Integration," *Proceedings of the IEEE*, 89(5), pages 602-633, May 2001.

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

[13] M. R. Becer, D. Blaauw, I. Algor, R. Panda, C. Oh, V. Zolotov, and I. N. Hajjy, "PostRoute Gate Sizing for Crosstalk Noise Reduction," *Proceeding of the ACM/IEEE Design Automation Conference,* pages 954-957, 2003.

[14] M. R. Becer, D. Blaauw, V. Zolotov, R. Panda, and I. N. Hajj, "Analysis of Noise Avoidance Techniques in DSM Interconnects using a Complete Crosstalk Noise Model," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition,* pages 456-463, 2002.

[15] M. Berkelaar, K. Eikland, and P. Notebaert, "LP_SOLVE: Open source (Mixed-Integer) Linear Programming System," Version 5.5, May 2005. (Available at http://groups.yahoo.com/group/lp_solve/)

[16] V. Betz, "VPR and T-VPack: Versatile Packing, Placement and Routing for FPGAs," Version 4.30, March 2000. (Available at http://www.eecg.toronto.edu/~vaughn/vpr/download.html)

[17] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," *Proceedings of the International Workshop on Field-Programmable Logic and Applications*, pages 213-222, 1997.

[18] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Near-Optimal Critical Sink Routing Tree Constructions," *IEEE Transactions on Computer-Aided Design,* 14(12), pages 1417-36, December 1995.

[19] C. C. Chang and J. Cong, "An Efficient Approach to Multilayer Layer Assignment with an Application to Via Minimization," *IEEE Transactions on Computer-Aided Design,* 18(5), pages 608-620, May 1999.

[20] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky, "Area Fill Synthesis for Uniform Layout Density," *IEEE Transactions on Computer-Aided Design,* 21(10), pages 1132-1147, October, 2002.

[21] Y. Chen, P. Gupta, and A. B. Kahng, "Performance-Impact Limited Area Fill Synthesis," *Proceeding of the ACM/IEEE Design Automation Conference,* pages 22-27, 2003.

[22] Y. H. Cheng and Y. W. Chang, "Buffer Planning: Integrating Buffer Planning with Floorplanning for Simultaneous Multi-Objective Optimization," *Proceedings of the IEEE Asia and South Pacific Design Automation Conference,* pages 624-627, 2004.

[23] Y. K. Cheng and S. M. Kang, "An Efficient Method for Hot-Spot Identification in ULSI Circuits," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 124-127, 1999.

[24] T. Y. Chiang, K. Banerjee, and K. C. Saraswat, "Effect of Via Separation and Low-k Dielectric Materials on the Thermal Characteristics of Cu Interconnects," *Digest of the IEEE International Electron Devices Meeting*, pages 261-264, 2000.

[25] T. Y. Chiang, K. Banerjee, and K. C. Saraswat, "Compact Modeling and SPICE-Based Simulation for Electrothermal Analysis of Multilevel ULSI Interconnects," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 165-172, 2001.

[26] P. Christie and D. Stroobandt, "The Interpretation and Application of Rent's Rule," *IEEE Transactions on Very Large Integration Systems,* 8(6), pages 639-648, December 2000.

[27] J. Cong, "Challenges and Opportunities for Design Innovations in Nanometer Technologies," *Proceedings of the International Symposium on Computing and Microelectronics Technologies,* 1998. (Available at: http://ballade.cs.ucla.edu/~ cong/papers/final1.pdf)

[28] J. Cong, "An Interconnect-Centric Design Flow for Nanometer Technologies," *Proceedings of the International Symposium on VLSI Technology, Systems, and Applications,* pages 54-57, 1999.

[29] J. Cong and Y. Ding, "Flowmap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Transactions on Computer-Aided Design,* 13(1), pages 1-12, January 1994.

[30] J. Cong, L. He, C. K. Koh, and P. H. Madden, "Performance Optimization of VLSI Interconnect Layout," *Integration: the VLSI Journal,* 21(1), pages 1-94, November 1996.

[31] J. Cong, T. Kong, and D. Z. Pan, "Buffer Block Planning for Interconnect-Driven Floorplanning," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design,* pages 358-363, 1999.

[32] J. Cong, D. Z. Pan, and P.V. Srinivas, "Improved Crosstalk Modeling for Noise Constrained Interconnect Optimization," *Proceedings of the IEEE Asia and South Pacific Design Automation Conference,* pages 373-378, 2001.

[33] J. Cong and Y. Zhang, "Thermal-Driven Multilevel Routing for 3-D ICs," *Proceedings of the IEEE Asia and South Pacific Design Automation Conference,* pages 121-126, 2005.

[34] J. Cong, Y. Zhang, and M. Xie, "*MARS* Multilevel Full-Chip Routing System," Version 1.0, 2001. (Available at: http://cadlab.cs.ucla.edu/~ pubbench/routing/)

[35] T. H. Cormen, C. E. Leiseron, R. L. Rivest, and C. Stein, *Introduction to Algorithms,* second edition, MIT Press, Cambridge, MA, 2001.

[36] W. Dai, L. Wu, and S. Zhang, "GSRC Floorplan Benchmark Suite." (Available at: http://www.cse.ucsc.edu/research/surf/GSRC/GSRCbench.html)

[37] S. Das, A. Chandrakasan, and R. Reif, "Design Tools for 3-D Integrated Circuits," *Proceedings of the IEEE Asia and South Pacific Design Automation Conference*, pages 53-56, 2003.

[38] J. A. Davis, V. K. De, and J. D. Meindl, "A Stochastic Wire-Length Distribution for Gigascale Integration (GSI) - Part I: Derivation and Validation," *IEEE Transactions on Electron Devices*, 45(3), pages 580-589, March 1998.

[39] Y. Deng and W. Maly, "Physical Design of the "2.5D" Stacked System," *Proceedings of the International Conference on Computer Design*, pages 211-217, 2003.

[40] A. Devgan, "Efficient Coupled Noise Estimation for On-Chip Interconnect," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design,* pages 147-151, 1997.

[41] W. E. Donath, "Wire length distribution for placements of computer logic," *IBM Journal of Research and Development,* 25(3), pages 152-155, May, 1981.

[42] F. F. Dragan, A. B. Kahng, I. Mandoiu, and S. Muddu, "Provably Good Global Buffering Using an Available Buffer Block Plan," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design,* pages 358-363, 2000.

[43] W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *Journal of Applied Physics,* 19(1), pages 55-63, January 1948.

[44] R. J. Enbody and K. H. Tan, "Routing the 3-D Chip," *Proceedings of the ACM/IEEE Design Automation Conference*, pages 132-137, 1992.

[45] M. R. Garey and D. S. Johnson, *Computers and Intractability,* Freeman Press, San Francisco, CA, 1979.

[46] A. V. Goldberg, "An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm", *Journal of Algorithms*, 22(1), pages 1-29, 1997.

[47] B. Goplen and S. S. Sapatnekar, "Efficient Thermal Placement of Standard Cells in 3D ICs Using a Force Directed Approach," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 86-89, 2003.

[48] B. Goplen and S. S. Sapatnekar, "Thermal Via Placement in 3D ICs," *Proceedings of the ACM International Symposium on Physical Design*, pages 167-174, 2005.

[49] P. Gupta and A. B. Kahng, "Manufacturing-Aware Physical Design," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design,* pages 681-687, 2003.

[50] R. Gupta, B. Krauter, B. Tutuianu, and L. T. Pileggi, "The Elmore Delay as a Bound for RC Trees with Generalized Input Signals," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 364-369, 1995.

[51] R. T. Hadsell and P. H. Madden, "Improved Global Routing through Congestion Estimation," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 28-34, 2003.

[52] N. Hanchate and N. Ranganathan, "Post-Layout Gate Sizing for Interconnect Delay and Crosstalk Noise Optimization," *Proceedings of the International Symposium on Quality Electronic Design,* pages 92-97, 2006.

[53] N. Hanchate and N. Ranganathan, "A Linear Time Algorithm for Wire Sizing with Simultaneous Optimization of Interconnect Delay and Crosstalk Noise," *Proceedings of the International Conference on VLSI Design,* pages 283-290, 2006.

[54] M. Hashimoto, M. Takahashi, and H. Onodera, "Crosstalk Noise Optimization by PostLayout Transistor Sizing," *Proceedings of the ACM International Symposium on Physical Design*, pages 126-130, 2002.

[55] L. He, A. B. Kahng, K. H. Tam, and J. Xiong, "Variability-Driven Considerations in the Design of Integrated-Circuit Global Interconnects," *Proceedings of the International VLSI Multilevel Interconnection Conference,* pages 214-221, 2004.

[56] L. He, A. B. Kahng, K. H. Tam, and J. Xiong, "Simultaneous Buffer Insertion and Wire Sizing Considering Systematic CMP Variation and Random Leff Variation," *Proceedings of the ACM International Symposium on Physical Design,* pages 78-85, 2005.

[57] L. He, A. B. Kahng, K. Tam, and J. Xiong, "Design of IC Interconnects with Accurate Modeling of CMP," *Proceedings of the SPIE Symposium on Microlithograhpy*, pages 109-119, 2005.

[58] P. Heydari and M. Pedram, "Capacitive Coupling Noise in High-Speed VLSI Circuits," *IEEE Transactions on Computer-Aided Design,* 24(3), pages 478-488, March 2005.

[59] B. Hu, H. Jiang, Q. Liu, and M. Marek-Sadowska, "Synthesis and Placement Flow for Gain-based Programmable Regular Fabrics," *Proceedings of the ACM International Symposium on Physical Design,* pages 197-203, 2003.

[60] J. Hu and S. S. Sapatnekar, "A Survey on Multi-Net Global Routing for Integrated Circuits," *Integration: The VLSI Journal,* 31(1), pages 1-49, November 2001.

[61] T. Y. Ho, Y. W. Chang, and S. J. Chen, "Multilevel Routing with Antenna Avoidance," *Proceedings of the ACM International Symposium on Physical Design,* pages 34-40, 2004.

[62] T. Jing, L. Zhang, J. H. Liang, J. Y. Xu, X. L. Hong, J. J. Xiong, and L. He, "A Min-Area Solution to Performance and RLC Crosstalk Driven Global Routing Problem," *Proceedings of the IEEE Asia and South Pacific Design Automation Conference,* pages 115-120, 2005.

[63] A. B. Kahng, S. Muddu, and D. Vidhani, "Noise and Delay Uncertainty Studies for Coupled RC Interconnects," *Proceedings of the IEEE International ASIC/SOC Conference,* pages 3-8, 1999.

[64] A. B. Kahng, G. Robins, A. Singh, and A. Zelikovsky, "Filling Algorithms and Analyzes for Layout Density Control," *IEEE Transactions on Computer-Aided Design*, 18(4), pages 445-462, April 1999.

[65] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "hMETIS-Hypergraph & Circuit Partitioning," Version 1.5.3, November 1998. (Available at: http://www-users.cs.umn.edu/~karypis/metis/hmetis/)

[66] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "An Exact Algorithm for Coupling-Free Routing," *Proceedings of ACM International Symposium on Physical Design,* pages 10-15, 2001.

[67] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling," *IEEE Transactions on Computer-Aided Design,* 21(7), pages 777-790, July 2002.

[68] R. Kay and R. A. Rutenbar, "Wire Packing: A Strong Formulation of Crosstalk-Aware Chip-Level Track/Layer Assignment with an Efficient Integer Programming Solution," *Proceedings of the ACM International Symposium on Physical Design,* pages 61-68, 2000.

[69] S. Khatri, A. Mehrotra, R. Brayton, and A. Sangiovanni-Vincentelli, "A Novel VLSI Layout Fabric for Deep Sub-Micron Applications," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 491-496, 1999.

[70] V. Kheterpal and L. Pileggi, "Routing Architecture Exploration for Regular Fabrics," *Proceedings of the ACM/IEEE Design Automation Conference*, pages 204-207, 2004.

[71] M. R. Kramer and J. van Leeuwen, "The Complexity of Wire Routing and Finding Minimum Area Layout for Arbitrary VLSI Circuits," in F. P. Preparata, editor, *Advances in Computing Research,* volume 2: VLSI theory, pages 129-146, JAI, Reading, MA, 1984.

[72] M. Kuhlmann and S. S. Sapatnekar, "Exact and Efficient Crosstalk Estimation," *IEEE Transactions on Computer-Aided Design*, 20(7), pages 858-866, July 2001.

[73] A. Kurokawa, T. Kanamoto, T. Ibe, A. Kasebe, C. W. Fong, T. Kage, Y. Inoue, and H. Masuda, "Dummy Filling Methods for Reducing Interconnect Capacitance and Number of Fills," *Proceedings of the International Symposium on Quality Electronic Design,* pages 586-591, 2005.

[74] B. S. Landman and R. L. Russo, "On a Pin Versus Block Relationship for Partitions of Logic Graphs," *IEEE Transactions on Computers,* C-20(12), pages 1469-1479, December 1971.

[75] S. Lee, T. F. Lemczyk, and M. M. Yovanovich, "Analysis of Thermal Vias in High Density Interconnect Technology," *Proceedings of the Semiconductor Thermal Measurement, Modeling, and Management Symposium*, pages 55-61, 1992.

[76] W. S. Lee, K. H. Lee, J. K. Park, T. K. Kim, Y. K. Park, and J. T. Kong, "Investigation of the Capacitance Deviation Due to Metal-Fills and the Effective Interconnect Geometry Modeling," *Proceedings of the International Symposium on Quality Electronic Design,* pages 373-376, 2003.

[77] S. M. Li, Y. H. Cheng, and Y. W. Chang, "Performance Driven Floorplan: Noise-Aware Buffer Planning for Interconnect-Driven Floorplanning," *Proceedings of the IEEE Asia South Pacific Design Automation Conference,* pages 423-426, 2003.

[78] P. Li, T. Pileggi, M. Asheghi, and R. Chandra, "Efficient Full-Chip Thermal Modeling and Analysis," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 319-326, 2004.

[79] J. Lillis, C. K. Cheng, and T. T. Y. Lin, "Optimal wire sizing and buffer insertion for low power and a generalized delay model," *Proceedings of IEEE/ACM International Conference on Computer-Aided Design,* pages 138-143, 1995.

[80] S. P. Lin and Y. W. Chang, "A Novel Framework for Multilevel Routing Considering Routability and Performance," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design,* pages 44-50, 2002.

[81] P. Madden, "MCNC Standard Cell Benchmark Circuits." (Available at: http://vlsicad.cs.binghamton.edu/~ pmadden/benchmark.html)

[82] P. B. Morton and W. Dai, "An Efficient Sequential Quadratic Programming Formulation of Optimal Wire Spacing for Crosstalk Noise Avoidance Routing," *Proceedings of the ACM International Symposium on Physical Design,* pages 22-28, 1999.

[83] P. B. Morton and W. Dai, "Crosstalk Noise Estimation for Noise Management," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 659-664, 2002.

[84] G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics,* 38(8), pages 114-117, April 1965.

[85] L. W. Nagel, "SPICE2, a Computer Program to Simulate Semiconductor Circuits," University of California, Berkeley, CA, Technical Report ERL-M520, May 1975.

[86] R. Nair, "A Simple Yet Effective Technique for Global Wiring," *IEEE Transactions on Computer-Aided Design,* 6(2), pages 165-172, March 1987.

[87] G. Nanz and L. E. Camilletti, "Modeling of Chemical-Mechanical Polishing: A Review," *IEEE Transactions on Semiconductor Manufacturing*, 8(4), pages 382-389, April 1995.

[88] A. Odabasioglu, M. Celik, and L. T. Pillage, "PRIMA: Passive Reduced Order Interconnect Macromodeling Algorithm," *IEEE Transactions on Computer-Aided Design,* 17(8), pages 645-654, August 1998.

[89] D. Ouma, D. Boning, J. Chung, G. Shinn, L. Olsen, and J. Clark, "An Integrated Characterization and Modeling Methodology for CMP Dielectric Planarization," *Proceedings of the IEEE International Interconnect Technology Conference*, pages 67-69, 1998.

[90] P. M. Pardalos and L. S. Pitsoulis, *Nonlinear Assignment Problems: Algorithms and Applications,* Kluwer Academic Publishers, Dordrecht, Holland, 2000.

[91] C. Patel, A. Cozzie, H. Schmit, and L. Pileggi, "An Architectural Exploration of Via Patterned Gate Arrays," *Proceedings of the ACM International Symposium on Physical Design,* pages 184-189, 2003.

[92] L. T. Pillage and R. A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Transactions on Computer-Aided Design,* 9(4), pages 352-366, April 1990.

[93] L. Pillage, R. Rohrer, and C. Visweswarish, *Electronic Circuit and System Simulation Methods,* McGraw-Hill, New York, NY, 1995.

[94] L. Pileggi, H. Schmit, A. J. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Y. Tong, "Exploring Regular Fabrics to Optimize the Performance Cost Trade-off," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 782-787, 2003.

[95] J. Qian, S. Pullela, and L. T. Pillage, "Modeling the effective capacitance for the RC interconnect of CMOS gates," *IEEE Transactions on Computer-Aided Design,* 13(12), pages 1526-1535, December 1994.

[96] Y. Ran and M. Marek-Sadowska, "On Designing Via-Configurable Cell Blocks for Regular Fabrics," *Proceedings of the ACM/IEEE Design Automation Conference*, pages 198-203, 2004.

[97] P. Saxena and S. Gupta, "Shield Count Minimization in Congested Regions," *Proceedings of the ACM International Symposium on Physical Design,* pages 78-83, 2002.

[98] P. Saxena, N. Menezes, P. Cocchini, and D. A. Kirkpatrick, "The Scaling Challenge: Can Correct-by-Construction Design Help," *Proceedings of the ACM International Symposium on Physical Design,* pages 51-58, 2003.

[99] C. W. Sham and E. F. Y. Young, "Routability Driven Floorplanner with Buffer Block Planning," *IEEE Transactions on Computer-Aided Design,* 22(4), pages 470-480, April 2003.

[100] N. Sherwani, *Algorithms for VLSI Physical Design Automation,* third edition, Kluwer Academic Publishers, New York, NY, 1999.

[101] D. Sinha and H. Zhou,"Gate-Size Optimization Under Timing Constraints for Coupling-Noise Reduction," *IEEE Transactions on Computer-Aided Design,* 25(6), pages 1064-1074, June 2006.

[102] W. Shi and Z. Li, "An O(nlogn) Time Algorithm for Optimal Buffer Insertion," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 580-585, 2003.

[103] J. Singh and S. S. Sapatnekar, "Topology Optimization of Structured Power/Ground Networks," *Proceedings of the ACM International Symposium on Physical Design,* pages 116-123, 2004.

[104] T. Skalicky, "LASPack: A Package for Solving Large Sparse Systems of Linear Equations," Version 1.12, January 1996. (Available at: `http://www.tu-dresden.de/mwism/skalicky/laspack/laspack.html`)

[105] B. E. Stine, D. S. Boning, J. E. Chung, L. Camilletti, F. Kruppa, E. R. Equi, W. Loh, S. Prasad, M. Muthukrishnan, D. Towery, M. Berman, and A. Kapoor, "The Physical and Electrical Effects of Metal-Fill Patterning Practices for Oxide Chemical-Mechanical Polishing Processes," *IEEE Transactions on Electron Devices*, 45(3), pages 665-679, March 1998.

[106] D. Stroobandt, "Analytical Methods for a Priori Wire Length Estimates in Computer Systems," Ph.D. dissertation, University of Ghent, Faculty of

Applied Sciences, November 1998, (translated from Dutch). (Available at http://escher.elis.ugent.be/publ/Edocs/DOC/P101_039.pdf)

[107] H. Su, J. Hu, S. S. Sapatnekar, and S. R. Nassif, "Congestion-Driven Codesign of Power and Signal Networks," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 64-69, 2002.

[108] R. Tian, D. F. Wong, and R. Boone, "Model-Based Dummy Feature Placement for Oxide Chemical-Mechanical Polishing Manufacturability," *IEEE Transactions on Computer-Aided Design,* 20(7), pages 902-910, July 2001.

[109] C. H. Tsai and S. M. Kang, "Cell-Level Placement for Improving Substrate Thermal Distribution," *IEEE Transactions on Computer-Aided Design,* 19(2), pages 253-266, February 2000.

[110] L. P.P.P. van Ginneken, "Buffer Placement in Distributed RC-Tree Networks for Minimal Elmore Delay," *Proceedings of the IEEE International Symposium on Circuits and Systems,* pages 865-868, 1990.

[111] A. Vittal and M. Marek-Sadowska, "Crosstalk Reduction for VLSI," *IEEE Transactions on Computer-Aided Design,* 16(3), pages 290-298, March 1997.

[112] A. Vittal, L. H. Chen, M. Marek-Sadowska, K. P. Wang, and S. Yang, "Crosstalk in VLSI Interconnections," *IEEE Transactions on Computer-Aided Design,* 18(12), pages 1817-1824, December 1999.

[113] J. Westra, C. Bartels, and P. Groeneveld, "Probabilistic Congestion Prediction," *Proceedings of the ACM International Symposium on Physical Design,* pages 204-209, 2004.

[114] K. Wu and Y. Tsai, "Structured ASIC, Evolution or Revolution," *Proceedings of the ACM International Symposium on Physical Design,* pages 103-106, 2004.

[115] J. Xiong and L. He. "Full-Chip Routing Optimization with RLC Crosstalk Budgeting," *IEEE Transactions on Computer-Aided Design*, 23(3), pages 366-377, March 2004.

[116] J. Xiong and L. He, "Extended Global Routing with RLC Crosstalk Constraints", *IEEE Transactions on Very Large Scale Integration Systems,* 13(3), pages 319-329, March 2005.

[117] T. Xue, E. S. Kuh, and D. Wang, "Post Global Routing Crosstalk Risk Estimation and Reduction," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design,* pages 302-309, 1996.

[118] X. Yang, "IBM-PLACE Benchmarks," Version 1.0. (Available at: http://er.cs.ucla.edu/benchmarks/ibm-place/)

[119] T. Yu, S. Chheda, J. Ko, M. Roberton, A. Dengi, and E. Travis, "A Two-dimensional Low Pass Filter Model for Die-level Topography Variation Resulting from Chemical Mechanical Polishing of ILD Films," *Digest of the IEEE International Electron Devices Meeting*, pages 909-912, 1999.

[120] P. Zarkesh-Ha, J. A. Davis, and J. D. Meindl, "Prediction of interconnect fan-out distribution using Rent's rule," *Proceedings of the IEEE International Workshop on System Level Interconnect Prediction,* pages 107-112, 2000.

[121] Y. Zhan and S. S. Sapatnekar, "Fast Computation of the Temperature Distribution in VLSI Chips Using the Discrete Cosine Transform and Table Look-up," *Proceedings of the Asia and South Pacific Design Automation Conference,* pages 87-92, 2005.

[122] T. Zhang and S. S. Sapatnekar, "Simultaneous Shield and Buffer Insertion for Crosstalk Noise Reduction in Global Routing," *Proceedings of the IEEE International Conference on Computer Design*, pages 93-98, 2004.

[123] T. Zhang and S. S. Sapatnekar, "Buffering Global Interconnect in Structured ASIC Design," *Proceedings of the Asian South Pacific Design Automation Conference,* pages 23-26, 2005.

[124] T. Zhang, Y. Zhan, and S. S. Sapatnekar, "Temperature-Aware 3D Global Routing," *Proceedings of the Asian South Pacific Design Automation Conference*, pages 309-314, 2006.

[125] T. Zhang and S. S. Sapatnekar, "Simultaneous Shield and Buffer Insertion for Crosstalk Noise Reduction in Global Routing," submitted for review.

[126] T. Zhang and S. S. Sapatnekar, "Buffering Global Interconnect in Structured ASIC Design," submitted for review.

[127] T. Zhang and S. S. Sapatnekar, "CMP-Aware Routing for Minimized Dummy Metal Fills," submitted for review.

[128] H. Zhou and D. F. Wong, "Global Routing with Crosstalk Constraints," *Proceedings of the ACM/IEEE Design Automation Conference,* pages 374-377, 1998.