NETWORK THEORY AND APPLICATIONS

Layout Optimization in VLSI Design

Bing Lu, Ding-Zhu Du and
Sachin S. Sapatnekar
Editors

Springer-Science+Business Media, B.V.

Layout Optimization in VLSI Design

Network Theory and Applications

Volume 8

Managing Editors:

Ding-Zhu Du, *University of Minnesota, U.S.A.*

and

Cauligi Raghavendra, *University of Southern California, U.S.A.*

Layout Optimization in VLSI Design

Edited by

Bing Lu and Ding-Zhu Du

Department of Computer Science and Engineering,
University of Minnesota, Minneapolis, MN , U.S.A.

and

Sachin S. Sapatnekar

Department of Electrical and Computer Engineering,
University of Minnesota, Minneapolis, MN , U.S.A.

SPRINGER-SCIENCE+BUSINESS MEDIA, B.V.

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4419-5206-6 ISBN 978-1-4757-3415-7 (eBook)
DOI 10.1007/978-1-4757-3415-7

Printed on acid-free paper

All Rights Reserved
© 2001 Springer Science+Business Media Dordrecht
Originally published by Kluwer Academic Publishers in 2001
Softcover reprint of the hardcover 1st edition 2001

No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the copyright owner.

Contents

Preface

Introduction

The exponential scaling of feature sizes in semiconductor technologies has side-effects on layout optimization, related to effects such as interconnect delay, noise and crosstalk, signal integrity, parasitics effects, and power dissipation, that invalidate the assumptions that form the basis of previous design methodologies and tools. This book is intended to sample the most important, contemporary, and advanced layout optimization problems emerging with the advent of very deep submicron technologies in semiconductor processing. We hope that it will stimulate more people to perform research that leads to advances in the design and development of more efficient, effective, and elegant algorithms and design tools.

Organization of the Book

The book is organized as follows. A multi-stage simulated annealing algorithm that integrates floorplanning and interconnect planning is presented in Chapter 1. To reduce the run time, different interconnect planning approaches are applied in different ranges of temperatures. Chapter 2 introduces a new design methodology – *the interconnect-centric design methodology* and its centerpiece, *interconnect planning*, which consists of physical hierarchy generation, floorplanning with interconnect planning, and interconnect architecture planning. Chapter 3 investigates a net-cut minimization based placement tool, *Dragon*, which integrates the state of the art partitioning and placement techniques.

Chapter 4 deals with the single-net global routing tree optimization problem under stringent timing constraints. It is shown that the use of non-Hanan Steiner nodes is necessary for the maximum sink delay minimization problem and the specified delay achievement problem, and techniques for solving these problems are developed. Chapter 5 turns to techniques for combining routing and delay optimization for two pin nets. Basic timing-driven maze routing, simultaneous routing and basic

timing-driven maze routing, and buffer insertion algorithms are described in this chapter. An efficient approach for table-based inductance extraction and its applications are described in Chapter 6. In addition, the chapter develops a formulation and an algorithm for simultaneous shield insertion and net ordering problem for interconnect synthesis of multiple RLC nets.

As IC interconnects become narrower and are integrated in tighter physical configurations, more accurate characterization and modeling of the interconnect and package is demanded. Chapter 7 outlines methods for accurately characterizing and modeling the interconnect, signal delay, crosstalk noise, and simultaneous switching noise.

Chapter 8 presents a set of experimental results which show the effects of floorplanning, number of levels of metals, and supply voltage on area/delay requirement in digital binary adders designs. This case study emphasizes the importance of global floorplanning and highlights the need of new floorplanning algorithms which consider the above parameters.

To the Professional

The wide and advanced topics in this book make it an excellent handbook for researchers on VLSI CAD designs, heuristic algorithms, and approximation algorithms. Each chapter is relatively self-contained, and you may focus on topics that are of the greatest interest. You will also find the extensive bibliography useful to find advanced material on a topic.

Acknowledgements

We are grateful to all our colleagues and friends who have contributed greatly to the quality of this book. We would like to thank all of you for your useful suggestions and constructive criticisms. Finally, we would also like to thank reviewers and authors of each chapter for their continued support and their work on this book.

Minneapolis, Minnesota
U.S.A.
April, 2001

Bing Lu
Ding-Zhu Du
Sachin S. Sapatnekar

Integrated Floorplanning and Interconnect Planning

Hung-Ming Chen
Department of Computer Sciences
The University of Texas at Austin, Austin, TX 78712
E-mail: hmchen@cs.utexas.edu

Martin D.F. Wong
Department of Computer Sciences
The University of Texas at Austin, Austin, TX 78712
E-mail: wong@cs.utexas.edu

Hai Zhou
Advanced Technology Group
Synopsys, Inc. Mountain View, CA 94043
E-mail: haizhou@synopsys.com

Fung-Yu Young
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, Hong Kong
E-mail: fyyoung@cse.cuhk.edu.hk

Hannah H. Yang
Strategic CAD Labs
Intel Corporation, Hillsboro, OR 97124
E-mail: hyang@ichips.intel.com

Naveed Sherwani
Strategic CAD Labs
Intel Corporation, Hillsboro, OR 97124
E-mail: sherwani@ichips.intel.com

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 1–18.
© 2001 *Kluwer Academic Publishers.*

Contents

Abstract

When VLSI technology enters the deep sub-micron era, communication between different components is significantly increased. Interconnect delay also becomes the dominant factor in total circuit delay. All these make it necessary to start interconect planning as early as possible. In this chapter, we propose a method to combine interconnect planning with floorplanning. Our approach is based on the Wong-Liu floorplanning algorithm. When the positions, orientations, and shapes of the cells are decided, the pin positions and routing of the interconnects are decided as well. We use a multi-stage simulated annealing approach in which different interconnect planning methods are used in different ranges of temperatures to reduce running time. A temperature adjustment scheme is designed to give smooth transistions between different stages of simulated annealing. Experimental results show that our approach performs well.

1 Introduction

With VLSI technology entering the deep sub-micron (DSM) era, devices are scaled down to smaller sizes and placed at an ever increasing proximity.

At the same time, with the increase of die dimensions, more functions are integrated into one chip. All these significantly increase the communication between different components, thus increasing the amount of interconnect on a chip. Moreover, the scaling down of fabrication geometry also makes interconnect delay a dominant factor in total circuit delay [3]. These trends make interconnect planning a neccessary step in DSM design [7].

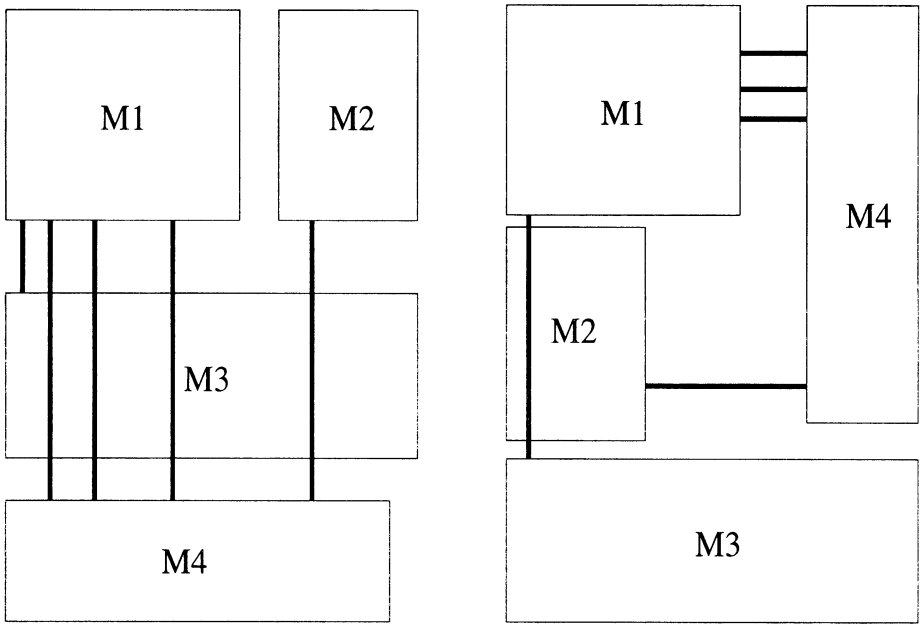


Figure 1: Floorplanning greatly influences interconnect structure

Global interconnects have significant influence on system performance in DSM technologies. Floorplanning, the process of placing functional blocks on the chip, can significantly affect the global interconnect structure. (Figure 1 shows two floorplans and their corresponding interconnect structures.) Many floorplanning algorithms have been proposed in the past 20 years [8, 9, 12, 5, 11, 6, 4]. All these algorithms focus on placing the circuit blocks using simple interconnect cost (e.g., total wire length) to guide the optimization. Without accurate interconnect planning during the floorplanning process, it is difficult for these algorithms to meet performance constraints due to unexpected "long" global interconnects resulted in the later routing stage.

In this chapter we propose a method to combine interconnect planning with floorplanning. Our approach is based on the Wong-Liu floorplanning

algorithm [12]. Recall that the Wong-Liu algorithm uses Polish expressions to represent floorplans and searches for an optimal floorplan using simulated annealing by iteratively generating Polish expressions. Every time a Polish expression (i.e., a floorplan) is examined, the shape of the blocks are optimized and the total wire length is used as the interconnect cost. Instead of using the total wire length, we propose to perform careful interconnect planning with respect to the current floorplan being considered and obtain a much more accurate interconnect cost. The comparison of the original approach and our new approach is shown in Figure 2.

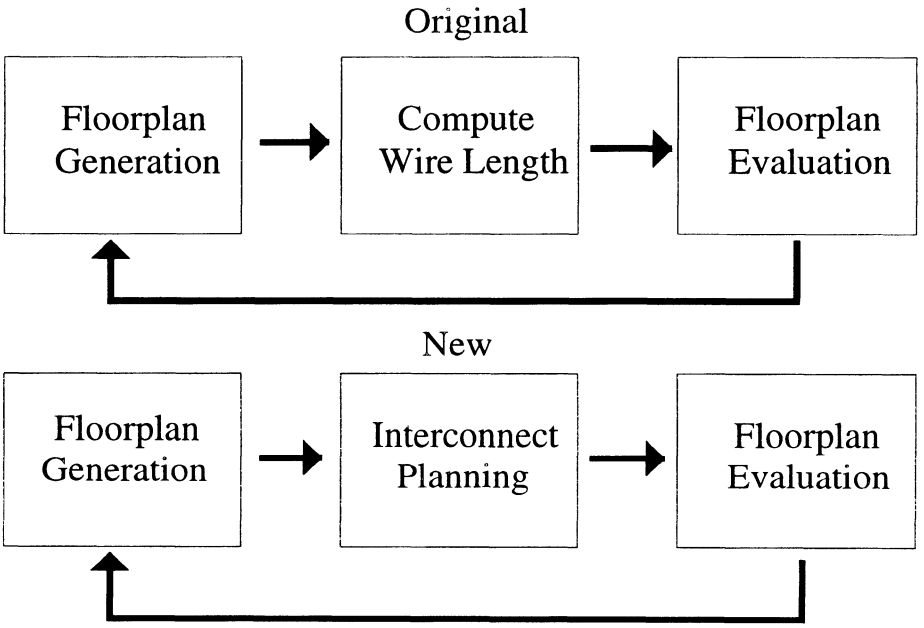


Figure 2: Floorplanning and interconnect planning

The interconnect planning step performs pin assignment and simple-geometry routing based on L-shaped and Z-shaped wires. Taking advantage of the nature of simulated annealing, we use different interconnect planning methods in different ranges of temperatures to reduce the running time. In particular, we use the conventional wire length estimation by half-perimeter of net bounding box when temperature is high, use a more accurate interconnect cost based on L-shaped routing when temperature is in the medium range, and finally use Z-shaped routing when temperature is low. In order to implement our multiple cost function scheme, we found that it was necessary to introduce a temperature adjustment method to cope with the

intrinsic discontinuities resulted in the process of switching cost functions.

The rest of the chapter is organized as follows. We introduce the algorithms for interconnect planning in Section 2. Section 3 discusses the multi-stage simulated annealing approach. Section 4 reports the experimental results for MCNC benchmarks and Section 5 concludes the chapter.

2 Efficient Interconnect Planning

To simplify our discussion, we assume there are two layers for the routing of global interconnects – one layer for vertical wires and the other layer for horizontal wires. (However, our approach is applicable to designs with more than two layers.) We allow different layers to have different design rules, i.e., the minimum wire width and the minimum spacing in each layer are different. In order to estimate congestion/routability, we divide the floorplan into a number of bins by a grid the same way that it is typically done in global routing [10]. For each bin boundary, we define its *capacity* as the maximum number of nets that can cross it. Clearly, the capacity of each boundary can be easily computed based on its length (or width) and the design rules (i.e., minimum wire width and minimum wire spacing) for that layer. If the number of nets crossing a bin boundary exceeds the capacity of the bin boundary, we say there is *overflow*. Each global routing solution gives us the number of nets crossing each bin boundary, thus giving us detailed congestion/overflow information. Our goal is to plan the interconnects to avoid congestion/overflow as much as possible.

2.1 Pin Assignment

The first step of interconnect planning is pin assignment. After module sizes and positions are fixed in a given floorplan, we determine the pin positions on each module. A simple strategy is used for efficiency. For each net, we connect the centers of the modules in this net and get the intersection points on the module boundaries as pin positions, as shown in Figure 3. This simple heuristic makes sense since it tries to minimize total wire length. Note that each module boundary is partitioned into a number of boundary segments by the grid. Since each boundary segment can only accomodate a limited number of pins, we should make sure that the number of pins we assign to each boundary segment does not exceed its capacity. If segment overflow occurs, we redistribute some of the pins to neighboring segments. Another guideline for pin assignment is to evenly distribute the pins so that no boundary segments are too crowded.

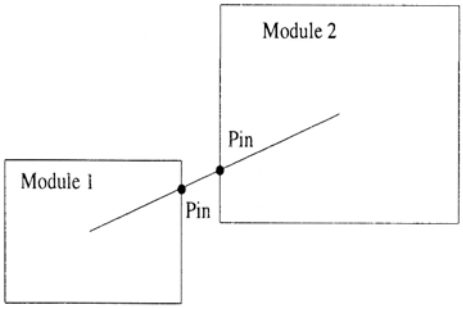


Figure 3: Pin assignment illustration

2.2 Simple-Geometry Routing

After pin assignment, pin positions are known. We then perform simple-geometry based global routing to connect the pins. For a net with n pins where $n > 2$, we first construct a minimum spanning tree connecting the pins using the Manhattan distance metric. The net is then decomposed into a set of two-terminal nets which correspond to the edges of the minimum spanning tree. After that, we have a set of nets with only two pins. For each of them, we connect the two pins using simple-geometry routing based on L-shaped or Z-shaped wires. Since the algorithms for L-shaped routing and Z-shaped routing are similar, they will be described together. Before we do simple-geometry routing, we map the pin positions of the nets to the corresponding bins. We use a sequential routing approach, that is, we route one net at a time. There are two steps in our simple-geometry routing algorithms. The first step is to use a stochastic approach to obtain the initial global congestion information. The second step is to utilize the information from the first step to route nets one by one.

In the first step, we estimate the congestion on each bin boundary by the expected number of nets crossing that boundary. Consider a two-pin net with pins $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. If only L-shaped routes are allowed, there are at most two routes to connect the two pins, as shown in Figure 4. Assume that each possible route is equally likely, we can add $1/2$ to each bin boundary on the two routes as the net's contribution to the expected number of nets crossing that boundary. For Z-shaped routing, we compute the expected number of nets crossing each bin boundary as follows. Let m denote the total number of Z-shaped routes connecting p_1 and p_2. As we can see, if $x_1 = x_2$ or $y_1 = y_2$, then $m = 1$. Otherwise, m can be

computed as follows.

$$m = |x_1 - x_2| + |y_1 - y_2|$$

For each bin boundary e, let m_e be the number of possible Z-shaped routes for the net to cross e. We again assume all routes are equally likely. Clearly, the net's contribution to the epxected number of nets crossing e is m_e/m. For the example shown in Figure 5 for Z-shaped routing, $m = 6$ and $m_e = 1$ where e is the right boundary of *bin(2,3)*. Thus the net's contribution to the expected number of nets crossing e is $m_e/m = 1/6$. Putting contributions from different nets together, we can get the expected number of crossing nets on each boundary.

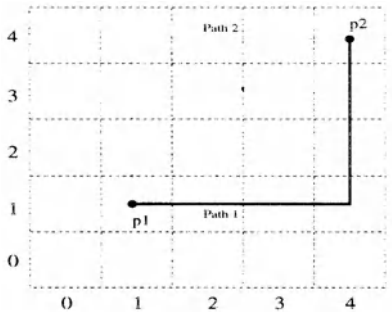


Figure 4: L-shaped routing

In the second step, we route one net at a time. When routing a net, we first remove its contribution from the expected number of crossing nets at each bin boundary. Then we determine a routing path with minimum crossing cost. The cost of crossing a bin boundary depends on a few factors. We use X_e to represent the overflow amount on bin boundary e. If there is no overflow on bin boundary e, let Y_e to be the difference between the current crossing and the capacity of e, and use Z to represent the overlapping length with previously routed wires belonging to the same (multi-pin) net. We determine a routing path which minimizes the following quantity: $\alpha \sum X_e^2 + \beta \sum 1/Y_e^2 - \gamma Z^2$. The first part is a *penalty* term, meaning that the global router is penalized because of going through the congested bin boundary. The second term is a *prevention* term, that is, the global router prevents from taking the path that is reaching saturation of the capacity. The third term is a *reward* that the router follows previous routes for those two-terminal nets within a multi-terminal net. After routing a net, if the route crosses a bin boundary e, its contribution to the expected number of nets crossing

e will become 1 to reflect the real route. If the current crossing of the bin boundary exceeds the capacity, mark this net to be riped-up and re-routed.

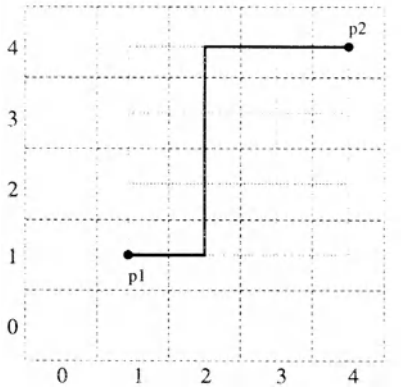


Figure 5: Z-shaped routing

For all nets that are needed to be riped-up and re-routed, we process them in the order from the most congested net, which is crossing the maximum number of congested bin boundaries, to the less congested ones trying to remove overflow as much as possible. Then we examine the results by getting the total square overflow terms of all bin boundaries. If the current overflow status exceeds the former one, recover the net to its original route.

2.3 Incremental Routing Cost Computation

A direct method to determine the path with minimum crossing cost connecting two points is as follows. For each possible path, L-shape or Z-shape, we need to sum up the crossing costs for all bin boundaries along the path to get the routing cost of this path. In this way, the time complexity of examining all L-shaped/Z-shaped paths joining two points is $O(n^2)$ in the worst case, where the grid size is n x n, since the total number of bin boundaries crossed by all L-shaped and Z-shaped paths between two points can be $O(n^2)$. It then follows that the total time to route N nets is $O(Nn^2)$.

In the following we present the idea of incremental routing cost computation which significantly speed up the cost function computation. For each (i, j), we define $v(i, j)$ as the accumulated crossing cost starting from the top bin boundary of $bin(i, 0)$ to that of $bin(i, j-1)$. Similarly, we define $h(i, j)$ as the accumulated crossing cost starting from the right bin boundary of $bin(0, j)$ to that of $bin(i-1, j)$. (See Figure 6, 7.) Note that all $h(i, j)$'s for a row can be computed in $O(n)$ time, and all $v(i, j)$'s for a column can be

computed in $O(n)$ time. Thus all $h(i,j)$'s and $v(i,j)$'s can be precomputed in $O(n^2)$ time.

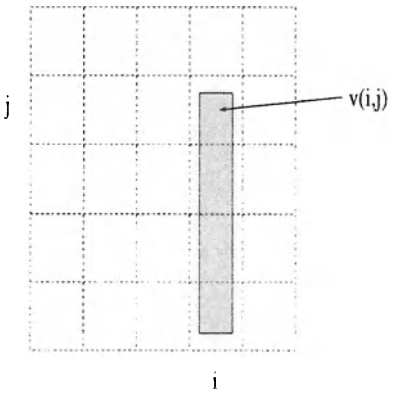


Figure 6: $v(i,j)$ is the accumulated cost of shaded bin boundaries

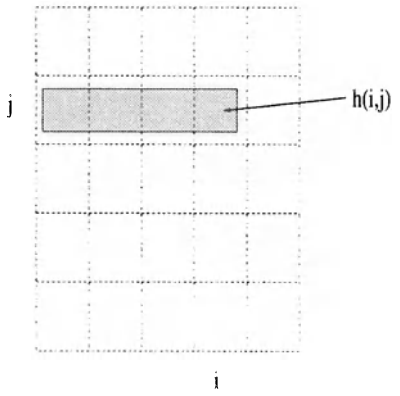


Figure 7: $h(i,j)$ is the accumulated cost of shaded bin boundaries

Note that the routing cost for each L-shaped/Z-shaped path can be expressed in terms of $h(i,j)$'s and $v(i,j)$'s. (The number of $h(i,j)$ or $v(i,j)$ terms in a L-shaped path is at most 4 and that for a Z-shaped path is at most 6.) For example, the routing cost of the Z-shaped path in Figure 8 can be computed by $v(3,3) - v(3,2) + h(3,2) - h(1,2) + v(1,2)$. So if all the $h(i,j)$'s and $v(i,j)$'s are precomputed, the time for evaluating all L-shaped/Z-shaped paths between two points is $O(n)$ since there are $O(n)$ such paths and the routing cost of each path can be computed in $O(1)$ time. After we route a path, we need to update the $h(i,j)$'s and $v(i,j)$'s on at most three columns/rows, and therefore can be done in $O(n)$ time. If there

are N nets, the total time for updating $h(i,j)$'s and $v(i,j)$'s is $O(Nn)$. As a result, the total time for routing N nets is $O(n^2 + Nn)$, which compares well with the $O(Nn^2)$ time direct method. The speed-up is roughly from cubic to quadratic in runtime.

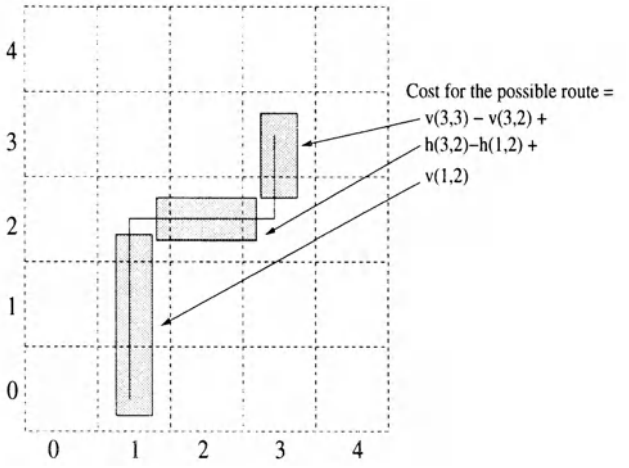


Figure 8: The routing cost in terms of $h(i,j)$'s and $v(i,j)$'s

3 Multi-Stage Simulated Annealing

Among our two interconnect planning approaches, Z-shaped routing is more accurate than L-shaped routing. But Z-shaped routing is also more expensive than L-shaped routing. Using Z-shaped routing all along will give the most accurate estimation. However, based on the characteristics of simulated annealing, we can speed up the procedure without sacrificing the quality of solutions.

The Wong-Liu floorplanning algorithm [12] is based on simulated annealing which is a technique for solving general optimization problems. The algorithm moves from one solution to another, trying to find the optimum solution. It accepts a move with the probability $e^{-\Delta C/T}$, where ΔC is the increase of cost by that move and T is the current temperature. When the temperature is very high, different estimation methods for the cost will not show much difference on $-\Delta C/T$. That means it does not affect much in performance if we use rough cost function at the beginning of annealing. When temperature gradually decreases, we use more accurate cost estimation. The L-shaped routing estimation is more accurate than the simple

center-to-center or half-perimeter estimation. Similarly, the Z-shaped routing is more accurate than the L-shaped routing estimation. Therefore, we will start with the the center-to-center or half-perimeter estimation, gradually transfer to L-shaped routing, and finally switch to Z-shaped routing. This multi-stage approach is very effective in reducing total running time.

In fact, multi-stage simulated annealing is just a method to combine different approaches together in one process. It should be reasonable if those different approaches used in multi-stage simulated annealing are not totally different, which means they have a certain degree of correlation. In this chapter we use a three-stage simulated annealing approach. The first stage is to get a good initial solution by using only the half-perimeter wire length estimation. The second stage is to estimate interconnect cost by using L-shaped global routing. The third stage is to estimate interconnect cost by using Z-shaped global routing. The transitions between stages are not very abrupt since they evolve from simple to complex, from rough to accurate. However, even for very similar estimations, we still need to find a way to take care of any possible discontinuity in switching cost functions.

3.1 Cost Function Transitions

The cost function used in [12] is $A + \lambda W$, where A is the total area of the packing, W is the half-perimeter estimation of the interconnect cost, and λ is a constant which controls the relative importance of these two terms and is usually set such that the area term and the interconnect term are approximately balanced. The normal curve of cost versus the number of iterations of simulated annealing process is shown in Figure 9.

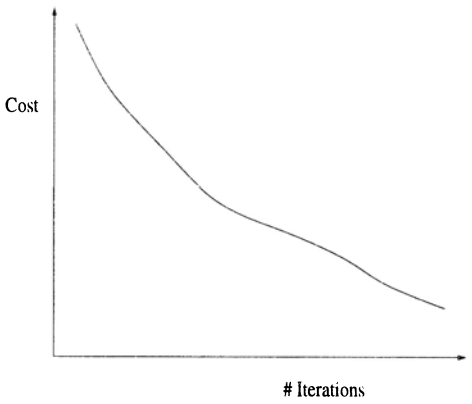


Figure 9: Normal curve of cost versus number of iterations

In our approach, we use the cost function $\Psi = \alpha A + \beta W + \gamma O$, where A and W are the same as in [12] and O is the sum of the square of overflow in routings. Although the format of cost function is identical for three stages of the process, the content of each term is different. The term W in stage 1 is obtained by applying half-perimeter method of net bounding boxes; the term W in stages 2 and 3 are obtained by applying pin assignment and summing the net length from pin positions. The term O in stage 1 is zero; the term O in stages 2 and 3 is obtained by applying simple-geometry routing and computing the congestion/routability estimation of bin boundaries. Because of the difference of cost functions used in different stages during simulated annealing process, discontinuities may occur when switching stages. One possible scenario is that the annealing process will suddenly converge to suboptimal solution when cost function transition occurs, as shown in Figure 10. The other possible scenario is that the annealing process will take much longer time to converge to optimal solution when cost function transition occurs, as shown in Figure 11. The discontinuities happen because the temperature is too low for the former scenario and is too high for the latter one when switching cost functions. In order to cope with the discontinuities resulted in the process of switching cost functions, we introduce a temperature adjustment method, which is described in the next sub-section.

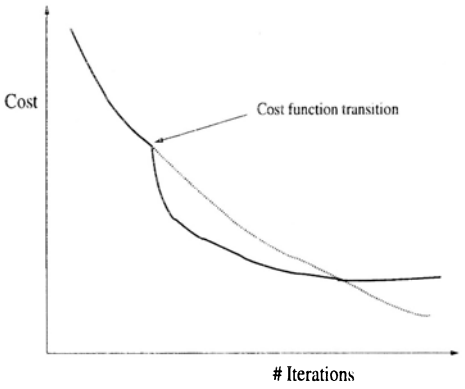


Figure 10: Abormal curve (solid line) of cost versus number of iterations: quickly converges to suboptimal solution

3.2 Temperature Adjustment

Simulated annealing uses temperature to control the probability in accepting uphill moves. We use a temperature schedule of the form $T_k = r * T_{k-1}, k =$

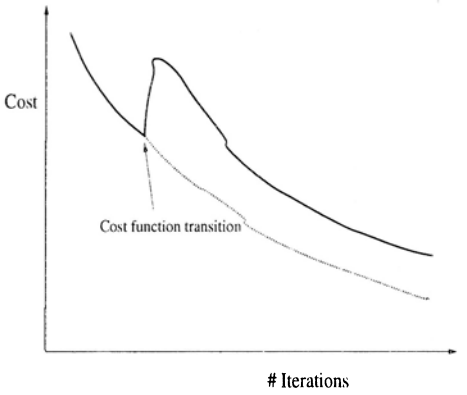


Figure 11: Abormal curve (solid line) of cost versus number of iterations: takes a much longer time to converge

$1, 2, 3, \ldots$. The initial temperature T_0 is determined by performing a sequence of random moves and computing the quantity Δ_{avg}, the average value of the magnitude of change in cost per move. We should have $e^{-\Delta_{avg}/T_0} = P \cong 1$ so that there will be a high probability of acceptance at high temperatures. This suggests that $T = -\Delta_{avg}/ln(P)$ is a good choice for T_0.

In [12], a single cost function is used to evaluate the quality of a solution. However, in our approach, we use different cost funcitons in different stages. We know that one major term to decide the acceptance of a solution in simulated annealing is $e^{-\Delta C/T}$. Take the transition between the first stage and the second stage as an example, the difference of cost in the second stage is typically larger than that in the first stage. That is, $-\Delta C_{old} >> -\Delta C_{new}$. Therefore, the probability of accepting uphill moves in the iterative-based process will decrease suddenly and the simulated annealing process will end prematurely. For example, when we encounter the stage transfer from half-perimeter estimation to L-shaped routing, suppose the current temperature is 100, the average ΔC_{old} is 20 and the average ΔC_{new} is 100. The probability of accepting uphill moves is $e^{-\Delta C_{old}/T} = 0.8$ before switching cost function but it is $e^{-\Delta C_{new}/T} = 0.36$ after cost function transition. This abrupt decrease in acceptance probability would result in quick convergence to suboptimal solution because the current temperature is too low to sustain the annealing process. Similarly, it is possible that after cost function transition, the acceptance probability will substantially increase. In this case, that the current temperature is too high results in slow convergence of the annealing process.

In our approach, in addition to calculating the starting temperature of

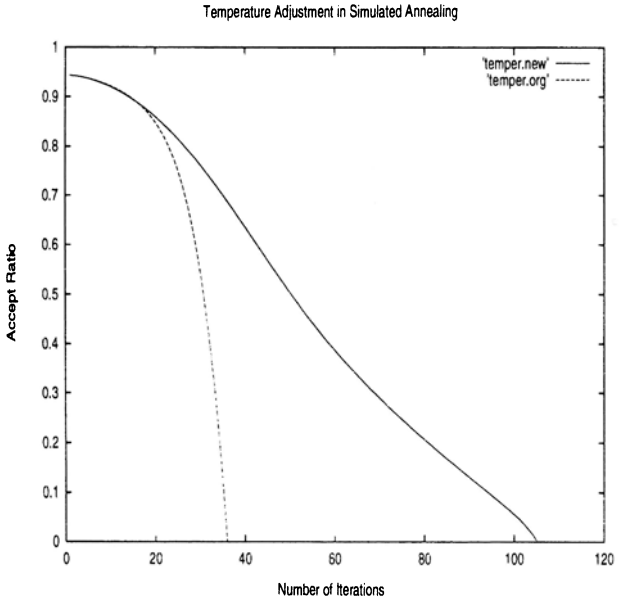


Figure 12: Temperature adjustment

the first stage, we also determine the starting temperature of the second and the third stages by calculating random move cost with the same approach. When we reach the transition between first and second stages or between second and third stages, we compute the starting temperature of the second or the third stage T' by getting the new average value of the magnitude of change in cost per move, and using the current acceptance ratio, P_{curr_acpt}, as a reference probability: $T' = -\Delta'_{avg}/ln(P_{curr\_acpt})$.

Although we use the current acceptance ratio to compute the new initial temperature during transition, the acceptance ratio will rise. The reason is that for the very first initial temperature estimation, we measure the term by random walks, but there exists very few random walk when transition occurs. We handle this by reducing the temperature much faster than the usual cooling ratio, until the acceptance ratio goes back on track. Experimental results show that this approach is really helping the continuity of the simulated annealing process and the quality of performance. (Figure 12 shows the effectiveness of applying temperature adjustment approach. The curve would have been the one in dotted line: it suddenly drops because of the abnormal end of the process.)

4 Experimental Results

We have tested our approach on some MCNC building blocks examples. All experiments were carried out on a 300MHz Pentium II Intel Processor. In order to compare the performance of the interconnect planning approach with that of the original approach [12] in terms of routability, we perform pin assignment and use Z-shaped routing to route the nets in the final floorplans produced by the conventional approach. Figure 13 shows the floorplan obtained by our pin assignment and interconnect planning approach. Figure 14 shows the floorplan obtained by the original approach. The dashed lines are the grid lines, and the thickness of line in the boundaries denotes the degree of overflow. We can see significant difference in Figure 13 and Figure 14 for *ami*49 benchmark in terms of wire overflow, while the packing areas are about the same (Table 1 and 2). For the five MCNC benchmarks shown in these two tables, we observe that the new approach produces floorplans which are much more routable than the ones produced by the original floorplanner. Note that the maximum violation in Table 1 indicates the maximum amount of overflow occurred in any bin boundary after interconnect planning, while the total violations indicate the total amount of overflow occurred in a floorplan. In fact, the new method achieves a significant percentage of improvement in maximum violation and total violations without any area overhead.

		Our Floorplanner				Floorplanner in [12]		
Data	n	Time (sec)	Dead Space(%)	Total Vios(μm)	Max Vio(μm)	Dead Space(%)	Total Vios(μm)	Max Vio(μm)
apte	9	277.6	0.99	0.51	0.27	0.86	10.31	3.45
xerox	10	589.7	0.14	0.0	0.0	0.07	23.92	8.88
hp	11	141.2	0.30	0.76	0.68	0.61	15.16	3.34
ami33	33	2220	3.66	1.55	0.64	5.68	15.96	2.64
ami49	49	4041	2.93	7.68	2.75	3.21	38.62	6.75

Table 1: Experimental results of our approach on MCNC examples, compared with the method in [9]

5 Conclusion

This chapter presents a method to integrate floorplanning with interconnect planning. Simple-geometry routing is used to efficiently plan wires during module packing. A congestion cost is combined into the Wong-Liu simulated annealing based floorplanner, and a multi-stage simulated annealing

			Improvement	
Data	n	#net	Total Vios(%)	Max Vio(%)
apte	9	97	95	92
xerox	10	203	100	100
hp	11	83	95	80
ami33	33	123	90	76
ami49	49	408	80	59

Table 2: Performance improvement of our approach on MCNC examples, compared with the method in [9]

strategy is used to effectively reduce the running time. We further develop a temperature adjustment approach to cope with the discontinuities resulting from switching cost functions. Experimental results show that our approach works well.

This chapter is an extended version of [1]. To further consider handling large number of nets and improving the quality of global interconnect routing, we refer readers to our recent work in [2].

6 Acknowledgements

This work was partially supported by the National Science Foundation under grant CCR-9912390 and by a grant from the Intel Corporation.

References

[1] H.-M. Chen, H. Zhou, F.Y. Young, D.F. Wong, H.H. Yang, and N. Sherwani, Integrated Floorplanning and Interconnect Planning, *IEEE International Conference on Computer-Aided Design*, pp. 354-357, 1999.

[2] H.-M. Chen, D.F. Wong, W.-K. Mak, and H.H. Yang, Faster and More Accurate Wiring Evaluation in Interconnect-Centric Floorplanning, *ACM Eleventh Great Lakes Symposium on VLSI*, pp. 62-67, 2001.

[3] J. Cong, L. He, K. Y. Khoo, C. K. Kuh, and Z. Pan, Interconnect Design for Deep Submicron, *IEEE International Conference on Computer-Aided Design*, pp. 478-485, 1997.

[4] P.-N. Guo, C.-K. Cheng, and T. Yoshimura, An O-Tree Representation of Non-Slicing Floorplan and its Applications, *ACM/IEEE Design Automaiton Conference*, pp. 268-273, 1999.

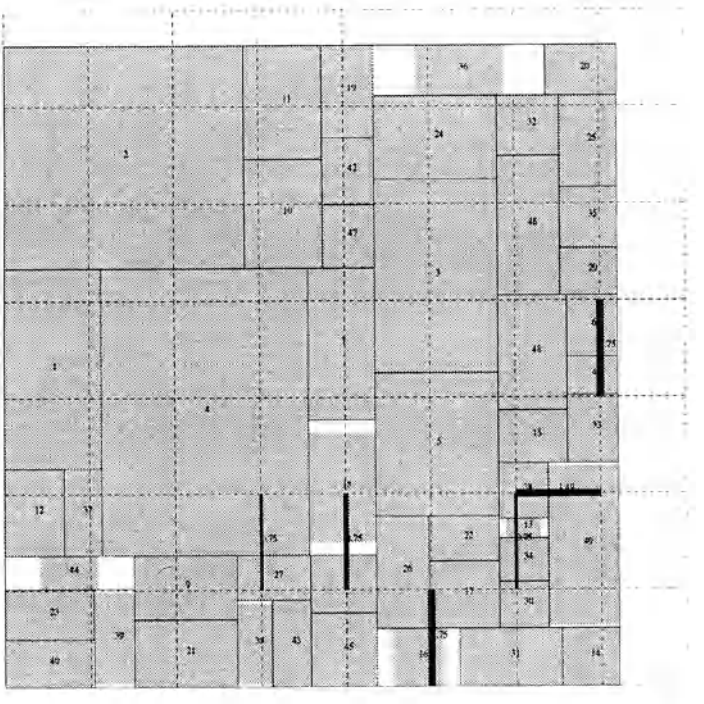


Figure 13: Result of packing ami49 using interconnect planning approach

[5] H. Murata, K. Fujiyoushi, S. Nakatake, and Y. Kajitan, Rectangle-Packing-Based Module Placement, *IEEE International Conference on Computer-Aided Design*, pp. 472-479, 1995.

[6] S. Nakatake, K. Fujiyoushi, H. Murata, Y. Kajitan, Module Placement on BSG-Structure and IC Layout Applications, *IEEE International Conference on Computer-Aided Design*, pp. 484-491, 1996.

[7] Ralph H.J.M. Otten and Robert K. Brayto, Planning For Performance, *ACM/IEEE Design Automaiton Conference*, pp. 122-127, 1998.

[8] R.H.J.M. Otten, Automatic Floorplan Design, *ACM/IEEE Design Automaiton Conference*, pp. 261-267, 1982.

[9] B. Preas and W. VanCleemput, Placement Algorithms for Arbitrary Shaped Blocks, *ACM/IEEE Design Automaiton Conference*, pp. 474-480, 1979.

[10] Naveed Sherwani, *Algorithms for VLSI Physical Design Automation*, (Boston, Kluwer Academic Publisher, 1995).

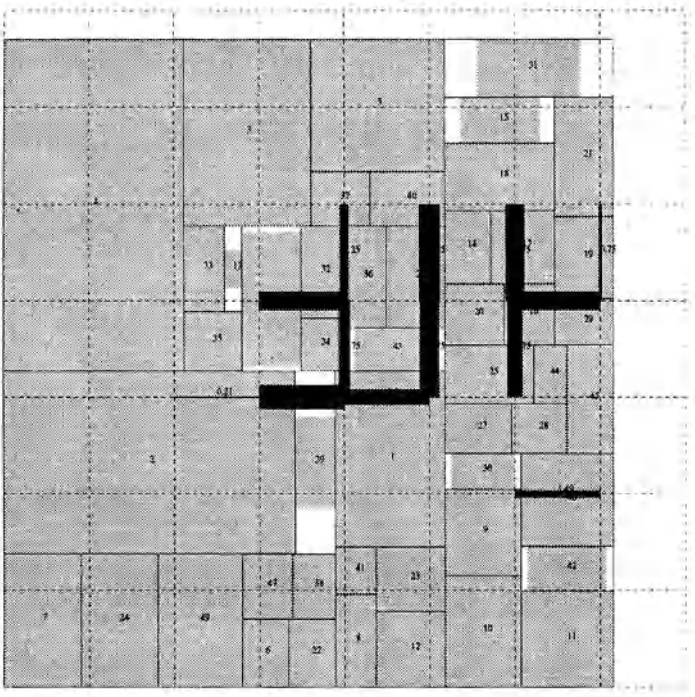


Figure 14: Result of packing ami49 using original approach

[11] T. Tamanouchi and K. Tamakashi and T. Kambe, Hybrid Floorplanning Based on Partial Clustering and Module Restructuring, *IEEE International Conference on Computer-Aided Design*, pp. 478-483, 1996.

[12] D.F. Wong and C.L. Liu, A New Algorithm for Floorplan Design, *ACM/IEEE Design Automaiton Conference*, pp. 101-107, 1986.

Interconnect Planning

Jason Cong
Department of Computer Science
University of California, Los Angeles, CA 90095
E-mail: cong@cs.ucla.edu

Contents

1 Introduction

The driving force behind the spectacular advancement of the integrated circuit technology in the past thirty years has been the *exponential scaling* of the transistor feature size, i.e., the minimum dimension of a transistor. It has been following the Moore's Law [1] at the rate of a factor of 0.7 reduction every three years. It is expected that such exponential scaling

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 19–44.
© 2001 *Kluwer Academic Publishers.*

will continue for at least another 10 to 12 years as projected in the 1997 National Technology Roadmap for Semiconductors (NTRS'97) [2] shown in Table 1.[1]

This will lead to over half a billion transistors integrated on a single chip with an operating frequency of 2 to 3 GHz in the 70nm technology by Year 2009.

Technology (nm)	250	180	150	130	100	70
Year	1997	1999	2001	2003	2006	2009
# transistors	11M	21M	40M	76M	200M	520M
Across chip clock (MHz)	750	1200	1400	1600	2000	2500
Area (mm^2)	300	340	385	430	520	620
Wiring Levels	6	6-7	7	7	7-8	8-9

Table 1: Overall technology roadmap from NTRS'97 [2].

With rapid feature size scaling, the circuit performance is increasingly determined by the interconnects instead of devices. The study in [4] computed the delays of a minimum size transistor, an average length interconnect (1mm), an un-optimized 2cm global interconnect, and an optimized 2cm global interconnect in each technology generation predicted in NTRS'97. It shows that although the intrinsic device delay of a minimum size transistor will decrease from 70ps in the 250nm technology down to about 20ps in the 70nm technology, the delay of an average interconnect (1mm metal line) will decrease only from about 60ps to 40ps, while the delays of a 2cm un-optimized global interconnect (with driver sizing only) will actually increase from about 2ns to 3.5ns. The delay of an optimized 2cm global interconnect is also computed after simultaneous driver sizing, buffer insertion, buffer sizing using the interconnect optimization package TRIO developed at UCLA [5]. Although such aggressive optimization reduces the 2cm global interconnect delay by 2× to 5× across different technology generations, it still does not reverse the trend of a growing gap between device and interconnect performance. It remains to be around 700ps, and is 20× and 30× that of

[1] NTRS'97 has been updated recently and the new version is called the 1999 International Technology Roadmap for Semiconductors (ITRS'99) [3]. The basic trend in ITRS'99 is the same as that in NTRS'97, although technology advancement is accelerated in ITRS'99 in certain areas. All the experimental results reported in this chapter are still based on NTRS'97.

a minimum size transistor in 100nm and 70nm technologies, respectively. [2] These data also implies that multiple clock cycles are needed for signals to travel over such optimized global interconnects for gigahertz designs in nanometer technologies. For example, even for a moderate clock frequency of 3GHz in the 70nm technology generation, 2 to 3 clock cycles are needed to travel through the 2cm optimized global interconnect. Note that this study has already considered the advances in the new interconnect materials as predicted in NTRS'97, with the use of copper at the 180nm generation and the use of low dielectric constant materials (the dielectric constant decreases from 3.55 in the 250nm technology to 1.5 in the 70nm technology). Although the use of these new interconnect materials is helpful in reducing interconnect delays, they do not provide the ultimate solution to the increasing performance mismatch between devices and interconnects. At best, they improve the interconnect performance by one or two technology generations, but the global interconnects remain the performance bottleneck. These results show clearly that the interconnect delay far exceeds the device delay and is the dominating factor in determining the system performance in current and future technology generations.

Signal reliability due to the coupling noise between interconnects is another serious problem in nanometer designs. In order to limit the increase of interconnect resistance, the wire aspect ratio (height over width) will increase considerably, from its current value of 1.8:1 in the 250nm technology to 2.7:1 in the 70nm technology as predicted in NTRS'97. The increase of wire aspect ratio together with the decrease of line-to-line spacing results in a rapid increase of coupling capacitance. The study in [4] shows that the coupling capacitance contributes to over 70% of the total capacitance under the minimum spacing and over 50% under two times (2×) the minimum spacing in all technology generations. As a result, even for such a moderate length wire of 1mm, with 2× the minimum width and spacing to its two neighbors, its peak noise reaches over 30% of Vdd in the 70nm generation [4]. Moreover, the value of crosstalk noise depends on not only the coupling capacitance of adjacent wires, but also the patterns and relative timing of the signals on neighboring wires. For example, under different switching patterns of neighboring wires, the noise value may differ by a factor of 2

[2]The interconnect process parameters provided in NTRS'97 are for a generic metal layer. It is likely that global interconnects will be put on higher metal layers, which have more aggressive reverse scaling. This may help to reduce the global interconnect somewhat. Although this reverse scaling was not considered in [4], it will not change the conclusion of the analysis.

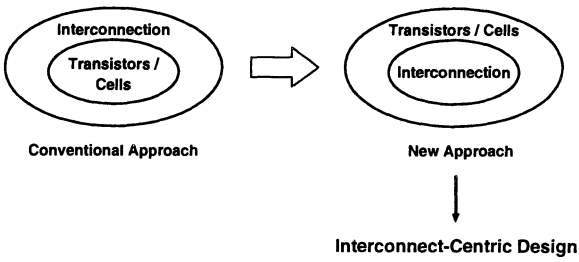


Figure 1: Proposed paradigm shift for interconnect-centric VLSI design.

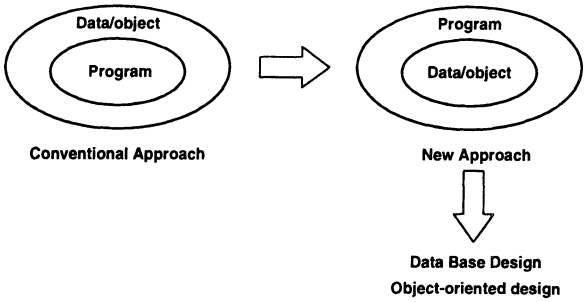


Figure 2: An analogous methodology change in software design.

to 3. For high-speed circuits, global interconnects may also be subject to inductive noise due to the coupling inductance of the interconnects. Both the capacitive and inductive noises due to the coupling of interconnects present serious threats to signal reliability in nanometer designs if they are not controlled properly.

Given the dominating importance of interconnects in current and future generations of IC designs, we have been developing a new design methodology, named *the interconnect-centric design methodology.* In conventional VLSI designs, much emphasis has been given to design and optimization of logic and devices. The interconnection was done by either layout designers or automatic place-&-route tools as an afterthought. In interconnect-centric designs, we suggest that interconnect design and optimization be considered and emphasized throughout the design process (see Figure 1). Such a paradigm shift is analogous to the one happened in the software design

domain of 1970s. In the early days of computer science, much emphasis was placed on algorithm design and optimization, while data organization was considered to be a secondary issue. It was recognized later on, however, that the data complexity is the dominating factor in many applications. This fact gradually led to a data-centric and object-centric software design methodology, including development of the database systems and the recent object-oriented design methodology (see Figure 2). Although algorithms and data representation/management are integral parts of any software system, the shift in viewpoint from algorithm-centric to data/object-centric designs allows us to effectively manage the design complexity in many large applications. We believe that development of the interconnect-centric design techniques and methodology will greatly impact VLSI designs, similar in the way that database design and object-oriented design methodologies have benefited software development.

Interconnect planning is the first step and also the centerpiece of our interconnect-centric design flow. It is applied very early on in the design process and has tremendous impact on the final result. We further divide the interconnect planning process into three steps: physical hierarchy generation, floorplanning with interconnect planning, and interconnect architecture planning. These are defined in the following paragraphs.

- *Physical hierarchy generation*: Designs in the nanometer technologies are inevitably hierarchical given their high complexity. However, the HDL description provided by the architecture and/or circuit designers usually follows the *logical hierarchy* of the design which reflects the logic dependency and relationship of various functions and components in the design. Such logical hierarchy may not map well to a two-dimensional layout solution as it is usually conceived with little or no consideration of the layout information. This is further evident from the sub-optimal results produced by many existing hierarchical design tools which use the logic hierarchy for floorplanning and recursive synthesis, placement and routing. Their results can be considerably worse than those by (good) flat design tools (when the design complexity is still tractable). Figure 3 shows an example of the logic hierarchy in the final layout (obtained by optimizing directly on the flat design). Modules in the same block in the logic hierarchy have the same grey shading in the layout. As can be seen, the logic hierarchy does not map directly into the physical hierarchy. This suggests that enforcing floorplanning or placement algorithms to follow the logic hierarchy boundary can be

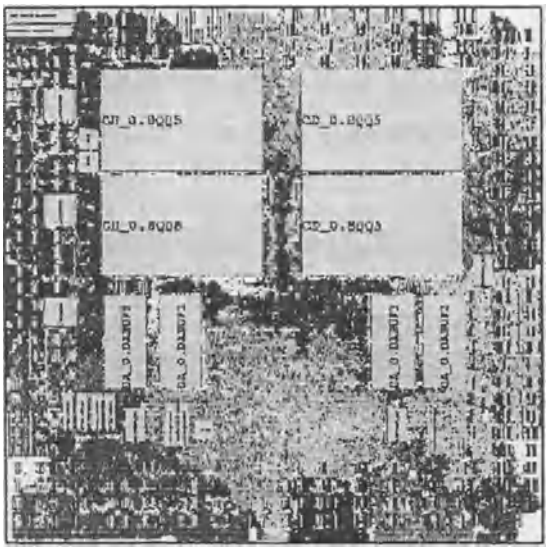


Figure 3: An example of logic hierarchy in the final layout (Courtesy of IBM). It is a large ASIC design with over 600,000 placeable objects, designed using IBM's SA27E technology (a 0.18μm technology with $L_{eff} = 0.11$ μm and using copper wires).

harmful to the final layout. Therefore, the first step of our interconnect planning process is to generate a good *physical hierarchy* that is most suitable for being embedded on a two-dimensional silicon surface for performance optimization. Such physical hierarchy generation in fact defines the global, semi-global, and local interconnects (based on their levels in the physical hierarchy) and has significant impact on the final design quality. In Section 2, we present our recent work on multi-level, multi-way, performance-driven partitioning with retiming as a possible approach to generating a good physical hierarchy. Retiming is considered during partitioning so that flip-flops can be repositioned onto the global interconnects to hide (some) global interconnect latency.

- Floorplanning with interconnect planning: After the physical hierarchy is generated, the second step is floorplanning with interconnect planning, which is also called *physical-level interconnect planning*. It

interacts closely with the interconnect synthesis tools and plans for the best interconnect topology, wire ordering, wire width and spacing, layer assignment, etc., for all global and semi-global interconnects to meet the required performance. For example, it is estimated that there will be a large number of buffers to be inserted for high-performance designs in future technology generations (close to 800,000 in 70nm technology [4]). If these buffers are distributed over the entire chip in an unstructured way, it will definitely complicate the layout design and verification. Section 3 presents a method to automatically plan for buffer blocks during floorplanning to achieve performance, area, and routability optimization.

- Interconnect architecture planning: Due to the advance in VLSI fabrication technology, such as the use of chemical-mechanical polishing (CMP) for global and local planarization of insulator and metal levels, the design rules are no longer completely dictated by the manufacturing capability and leave large room for optimization. The goal of *interconnect architecture planning* is to take advantage of the degree of freedom in the process technology and determine various interconnect parameters for overall system-level performance, reliability and power optimization, subject to the manufacturing constraints. These parameters include the number of routing layers, the thickness of each interconnect and isolation layer, the metal resistivity and dielectric constant of each layer (assuming different material/process may be used for different layers for performance, yield, and cost considerations), the nominal width and spacing in each layer, vertical interconnection schemes (e.g., via dimensions and structures), and so on. Such interconnect architecture planning should consider a given design characterization (specified in terms of the target clock rate, interconnect distribution, depths of the logic network, etc.) obtained after physical hierarchy generation and floorplanning with interconnect planning. In some cases, such optimization requires adjustments in the fabrication process, which is more suitable and economical for high-volume designs (such as microprocessor designs) or a class of designs with similar design characterizations. We present in Section 4 our work on wire-width planning as an example of interconnect architecture planning, whose objective is to predetermine a small number of common wire widths in each layer so that they can be used for optimizing interconnects of a wide range of lengths in that layer.

The next three sections highlights our progress on physical hierarchy generation, floorplanning with interconnect planning, and interconnect architecture planning. In Section 5, we present how these interconnect planning techniques are used in our interconnect-centric design flow.

2 Performance-Driven Partitioning with Retiming

As we explained in the previous section, the first, and probably the most important step of interconnect planning process, is to transform the logic hierarchy implied in the design specification into a good physical hierarchy so that it is most suitable for being embedded on a two-dimensional silicon surface for performance optimization. We believe that this step can be achieved using (possibly recursive) partitioning and floorplanning/coarse placement with careful consideration of the impact on interconnect performance. Traditionally, partitioning is viewed and used as a mean to enable the divide-and-conquer methodology to tackle the design complexity (as used, for example, in the min-cut based placement approach). In our interconnect-centric design flow, however, we view top-down partitioning as a step that *defines* the interconnects — the connections between different blocks resulted from top-level partitioning become global interconnects, and the connections within the same block after several steps of partitioning become local interconnects. After applying partitioning recursively (sometimes together with coarse placement), we can define a hierarchy of interconnects, which in turn defines the *physical hierarchy* of the given design. In order to achieve this objective, we have developed a *performance-driven* partitioning algorithm with consideration of retiming. Our algorithm, named HPM, is different from the conventional partitioning algorithms in two ways.

- The HPM algorithm is targeted for performance optimization. Most conventional partitioning algorithms consider only cutsize minimization. Although this tends to minimize the total number of global interconnects, it does not consider the impact of the partitioning result on the overall circuit performance. For example, it is not desirable to have multiple global interconnects in a timing-critical path (recall that shows that the delay of 2cm global interconnect is about 10X to 20X larger than that of a 1mm local interconnect [4]). Yet the conventional partitioning algorithms make no effort to avoid such configuration.

- The HPM algorithm considers retiming during partitioning to hide (some) global interconnect latency. The benefit of considering retiming during partitioning can be illustrated by using a simple motivational example shown in Figure 4. The two partitioning solutions (a) and (b) both have the same cutsize of 1 and delay of 4, assuming that each node delay is 1, the intra-block connection delay is 0, and the inter-block connection delay is 2. The critical paths are shown in thick lines. Let us apply optimal retiming to both solutions. For solution (a), retiming cannot help to reduce the delay (see Figure 4 (c)). However, retiming can reduce the delay of solution (b) from 4 to 3 by repositioning of the flip-flops to hide part of the large inter-block delay (see Figure 4 (d)). This example suggests clearly that we can hide some global interconnect delay latency with proper consideration of retiming as we define the global interconnects during partitioning. We would like to emphasize that retiming over global interconnects is especially important to multi-giga hertz designs. As we pointed in Section 1, multiple clock cycles are needed to cross a global interconnect for multi-gigahertz designs in nanometer technologies. This can only be achieved with retiming and pipelining on global interconnects in synchronous designs. [3]

Given a sequential circuit, the HPM algorithm computes a partitioning solution with the minimum clock period under retiming with possible node replication. The area of each block in the partitioning solution is bounded by a given number A. For delay computation, the HPM algorithm assumes each gate v has a delay of d_v, each global interconnect between blocks has a delay of D, and each local interconnect delay within each block is 0.[4] The computation of the minimum clock period is achieved by solving a sequence of the decision problem formulated as follows: For a sequential circuit with a given target clock period ϕ and a given area bound on each block, decide if there exists a partitioning solution with a clock period of no more than ϕ

[3]Asynchronous design has the potential not to be limited by the global interconnect delay. In particular, a globally asynchronous, locally synchronous (GALS) is a promising design method that is currently being studied by the researchers in the Gigascale Silicon Research Center (GSRC). The detailed discussion of this topic is beyond the scope of this chapter. The reader may refer to related publications at http://www.gigascale.org for more details.

[4]This simplification is based on the fact that we lump the average local interconnect delay into the node delay d_v, based on the assumption that local interconnect delays do not vary much.

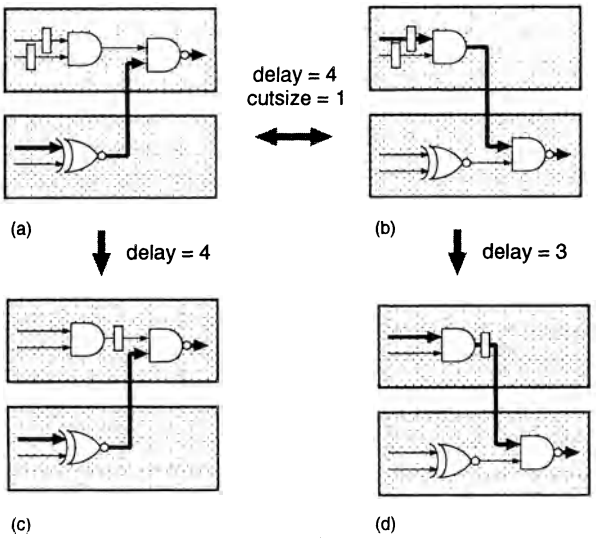


Figure 4: Advantage of simultaneous partitioning and retiming for delay minimization. Critical paths are shown in thick lines.

under the given delay model after retiming and possible logic replication.

The HPM algorithm integrates several recent advances in circuit partitioning in developing a highly efficient performance-driven partitioning algorithm with retiming. The concepts and techniques used in HPM include:

- The iterative label computation technique to test the feasibility of a proposed clock period under simultaneous partitioning and retiming [6].
- The highly efficient label computation procedure based on the monotone property of the label computation and the efficient longest path computation [7].
- The multi-level partitioning paradigm, which has led the best cutsize minimization based partitioning package hMETIS [8]
- An efficient performance-driven clustering algorithm (PRIME) with retiming [7].
- An efficient multi-level clustering algorithm based on global edge separability for cutsize minimization [9].

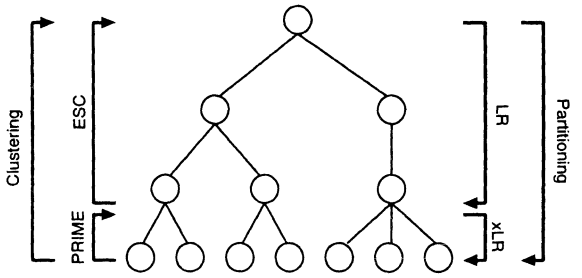


Figure 5: Main flow of the HPM algorithm along with the illustration of its multi-level cluster hierarchy.

- The multiway partitioning framework [10] to overcome the limitation of the recursive bipartitioning approach.

In particular, the HPM algorithm employs the multi-level partitioning framework as shown in Figure 5. Over the past twenty years, multilevel methods have been studied extensively as a means of accelerating numerical algorithms for partial differential equations [11, 12]. Application area are quite diverse, including image processing, combinatorial optimization, control theory, statistical mechanics, quantum electrodynamics, and linear algebra. Multilevel techniques for VLSI physical designs are currently an area of intensive research activity. Much progress has been made in multi-level circuit partitioning and placement. hMETIS [13] produces the best cut size minimization in circuit partitioning, and mPL [14] achieves competitive circuit placement with over 10× speed-up on designs with over 200K movable objects. The use of multi-level approach makes it feasible to extract the physical hierarchy from the flat design (which may consist of tens of millions of gates resulted from flattening the logic hierarchy).

The multi-level method is used in the HPM algorithm as follows. During the clustering phase of the HPM algorithm, a performance-driven clustering method with consideration of retiming [7] is used to build the base-level clustering structure to ensure the best possible subsequent retiming. Then it builds a multi-level clustering structure based on the global edge separability metric for cutsize minimization [9]. During the refinement phase of the HPM algorithm, simultaneous cutsize and performance-driven partitioning [15] is performed. It adopts a recently developed multiway partitioning framework

[10] to overcome the limitation of the recursive bipartitioning approach. As a result, the HPM algorithm produces partitioning solutions that are (a) 7% to 23% better in terms of delay compared to the best-known cutsize-driven hMETIS algorithm [8] with 19% increase in cutsize, and (b) 81% better in terms of cutsize compared to the best-known delay-driven PRIME algorithm [7] with only a 6% increase in delay.

Using the HPM algorithm, we generate the physical hierarchy as follows. Given a complex design specified in some HDL description (say either VHDL or Verilog language) in a hierarchical representation, we shall first perform a quick RTL synthesis and flatten the functional hierarchy as much as possible, down to a netlist of simple functional units (such as adders or decoders, but not necessarily gates) and pre-designed IP blocks. Then, we shall apply either the 2-way HPM algorithm recursively or the multi-way HPM algorithm with coarse placement to generate a good physical hierarchy for subsequent interconnect planning and synthesis steps.

The delay model used in HPM is simplistic as it assumes that all global interconnects have a uniform delay D. This assumption is due to the lack of physical information of the blocks generated by the HPM algorithm. Currently, the HPM algorithm is being extended to perform coarse placement/floorplanning together with partitioning and retiming [16]. It shows that the coarse placement operation can be naturally integrated into the multi-level framework used in the HPM algorithm. The placement information provides much more accurate global interconnect delay estimation and allows the possibility of repositioning flip-flops to the *middle* of a long global interconnect (not just at its two ends). Experimental results show that combining partitioning, coarse placement, and retiming can provide an additional 23% delay reduction compared to the physical planning results obtained by separate performance-driven partitioning followed by floorplanning. Given the efficiency and quality produced by the HPM algorithm (especially when combined with coarse placement), we believe that it is capable of extracting good physical hierarchies for large-scale designs in nanometer technologies.

3 Buffer Block Planning

After the physical hierarchy is generated, the second step of the interconnect planning process is floorplanning with interconnect planning. Traditional floorplanning algorithms focus on dimensions and placement of functional

blocks but ignore the interconnects associated with the design. We believe that floorplanning needs to interact closely with the interconnect synthesis tools to plan for the best interconnect topology, wire ordering and width, wire spacing, layer assignment, etc., for all global and semi-global interconnects in order to achieve the best possible circuit performance. As an example, we present here our recent work on automatic buffer block generation during floorplan design [17].

Technology(nm)	250	180	130	100	70
#buffers per chip	5K	25K	54K	230K	797K

Table 2: The number of buffers estimated for a high-performance design in each technology generation.

As shown in many recent studies (such as [18]), buffer insertion is a very effective technique to reduce the delay of long interconnects. Table 2 shows the number of buffers estimated for a high-performance design in each technology generation [4]. It is estimated that close to 800,000 buffers will be inserted in high-performance designs in the 70nm technology [4]. If so many buffers are arbitrarily distributed over a chip, it may cause several problems: (a) it makes it difficult to use/reuse pre-designed IP blocks, (b) it may complicate global/detailed routing and power/ground distribution, and (c) it may result in excessive area increase without proper planning. To overcome these problems, we propose to group buffers into buffer blocks. We have formulated the following *buffer block planning* (BBP) problem: Given an initial floorplan and the performance constraints for each net, we want to determine the optimal locations and dimensions of the buffer blocks such that the overall chip area and the number of buffer blocks after buffer insertion are minimized, while the performance constraint for each net is satisfied (assuming that it can be met by optimal buffer insertion). The output from our buffer block planning consists of the number of buffer blocks, each buffer block's area, location, and corresponding nets that use some buffer in this buffer block to meet the delay constraints.

Our study first shows that given a two-pin net, the *feasible region* (FR) for a buffer B, defined to be the maximum region where B can be located while still meeting the delay constraint, is quite large. The feasible region of a buffer can be computed according to the following theorem.

Theorem 3.1 *Given the route from the source to the sink, for a given delay constraint T_{req}, the feasible region $[x_{min}, x_{max}]$ for inserting one buffer is*

$$x_{min} = MAX\left(0, \left(K_2 - \sqrt{K_2^2 - 4K_1K_3}\right)/2K_1\right) \quad (1)$$

$$x_{max} = MIN\left(l, \left(K_2 + \sqrt{K_2^2 - 4K_1K_3}\right)/2K_1\right) \quad (2)$$

where

$$\begin{aligned} K_1 &= rc \\ K_2 &= (R_b - R_d)c + r(C_L - C_b) + rcl \\ K_3 &= R_dC_b + T_b + R_b(C_L + cl) + 1/2rcl^2 + rlC_L - T_{req}, \end{aligned}$$

r is the unit length wire resistance, c is the unit length wire capacitance, T_b is the intrinsic delay for the buffer, C_b is the input capacitance of the buffer, and R_b is the output resistance of the buffer. □

The proof of this theorem is available from [19]. Note that for Eqn. 1 and 2 to be valid, $K_2^2 - 4K_1K_3 \geq 0$ shall hold. Otherwise, no feasible region exists, and the initial floorplanning/timing budget has to be modified. Figure 6 shows the FR for inserting one buffer to an interconnect of length from 6mm to 9mm in the $0.18\mu m$ technology specified in NTRS'97. We first compute the best delay T_{best} by inserting one buffer, then set the delay constraint to be $(1 + \delta)T_{best}$, with δ varying from 0 to 50%. The x-axis shows the value of δ and the y-axis shows the length of the corresponding FR, i.e., $x_{max} - x_{min}$. It is interesting to see that even with a fairly small amount of slack, say 10% of T_{best}, the FR can be as much as 50% of the total wirelength!

When the route from the source to the sink is not specified, the feasible region is not just an interval, but a two-dimensional region which is the *union* of the one-dimensional feasible regions of *all* possible routes from source to sink. The *optimal* buffer locations, in this case, form a line segment of slope +1 or -1, for buffer insertion. Figure 7 shows an example of *a two-dimensional feasible region* with some routing obstacles. Obviously, routing obstacles need to be deducted from the feasible region computation.

When multiple buffers between the source and the sink of a net are needed to meet the delay constraint, we can compute the feasible region of each buffer again using an analytical formula similar to that in Eqn. 1 and 2, assuming that all other buffers are taking their optimal positions [17].

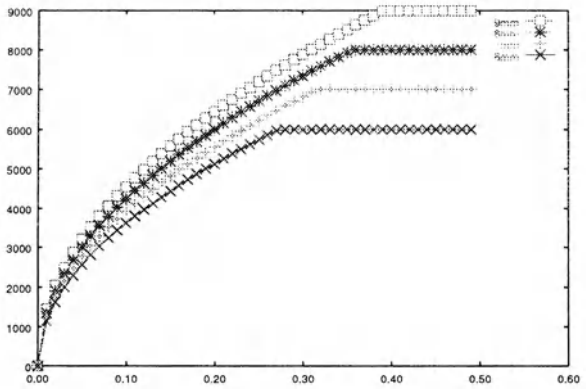


Figure 6: The distance of feasible region for inserting a buffer under different delay constraints specified by δ for length 6mm to 9mm wires in the $0.18\mu m$ technology. The x-axis shows the value of δ and the y-axis shows the length of the corresponding feasible region.

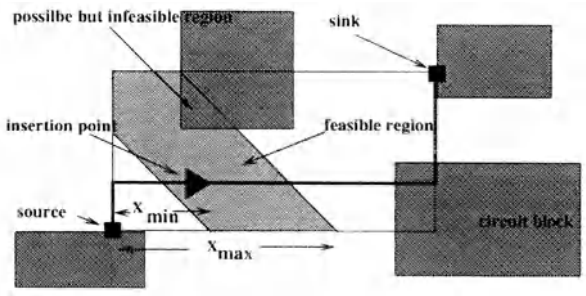


Figure 7: Two-dimensional feasible region. The existing circuit blocks act as obstacles for buffer insertion.

Theorem 3.2 *For a long interconnect with k buffers inserted, the feasible region for the i-th buffer ($i \leq k$) is $x_i \in [x_{min}(k,i), x_{max}(k,i)]$ with*

$$x_{min}(k,i) = MAX\left(0, \frac{K_2' - \sqrt{K_2'^2 - 4K_1'K_3'}}{2K_1'}\right)$$

$$x_{max}(k,i) = MIN\left(l, \frac{K_2' + \sqrt{K_2'^2 - 4K_1'K_3'}}{2K_1'}\right)$$

where K_1', K_2' and K_3' are functions of k and i (for simplicity of notation, we drop them in the above equations) with

$$\begin{aligned}
K_1'(k,i) &= \frac{(k+1)rc}{2i(k-i+1)} \\
K_2'(k,i) &= \frac{(R_b - R_d)c}{i} + \frac{r(C_L - C_b) + rcl}{k-i+1} \\
K_3'(k,i) &= kT_b - T_{req} + \left[R_d + (i-1)R_b + \frac{(k-i)rl}{k-i+1}\right] \cdot C_b \\
&+ R_b[(k-1)C_b + C_L + cl] + \frac{rcl^2}{2(k-i+1)} + rlC_L \\
&- \frac{(i-1)c(R_b - R_d)^2}{2ir} - \frac{(k-i)r(C_b - C_L)^2}{2(k-i+1)c}.
\end{aligned}$$

□

In this case, however, after a buffer is placed (i.e., "committed") to a position within its feasible region, we need to update the feasible regions of all other *unplaced* buffers of the same net to safely meet its delay constraint. Since we have an analytical formula, this update can be computed in constant time. A more recent study suggests a way to compute more conservative feasible regions to allow multiple buffers to be placed or moved simultaneously without violating the performance constraints [20].

Given the efficient procedures for computing feasible regions for buffer insertion, our buffer block planning algorithm works as follows. First, it builds the horizontal and vertical polar graphs of the given floorplan to keep track of available space for buffer insertion. The available space is divided into tiles. An *area slack* is computed for each tile, which measures the impact on the overall chip area if the tile area is increased. The algorithm iteratively chooses the tile with the maximum area slack and inserts a buffer

with the least flexibility (i.e., the minimal feasible region) into this tile. The area slacks of tiles and the feasible regions of the affected buffers (for multiple-buffer nets) may need to be updated after each buffer assignment. It was shown in [17] that this simple buffer planning scheme works well, mainly due to the large degree of freedom from feasible regions for buffer insertion. Experimental results show that the proposed algorithm can reduce the number of buffer blocks by a factor of 2.4× with smaller chip area and a better chance of meeting timing constraints and smaller overall chip area.

4 Wire Width Planning

After physical hierarchy generation and floorplanning with interconnect planning, we know the wirelength distribution on each layer. In this case, it is possible to perform interconnect architecture planning for each layer for optimizing the performance and cost of the overall design as discussed in the beginning of Section 1. This section presents our results on wire width planning, which is part of our overall effort on interconnect architecture planning. As shown in a number of recent studies (see [18] for a summary), wire sizing is an effective technique for reducing interconnect delays. However, having many different wire widths will considerably complicate the layout design, especially the routing process. Therefore, it is interesting to investigate the possibility of using a small set of predetermined "fixed" widths in each layer to get close to optimal performance for all interconnects in a wide range of wirelengths in that layer (not just one length).

Given the wirelength distribution in each layer (which can be obtained accurately after floorplanning with interconnect planning), the *wire-width planning problem* is to find the best width vector $\vec{W}$ for that layer such that the following objective function

$$\Phi(\vec{W}, l_{min}, l_{max}) = \int_{l_{min}}^{l_{max}} \lambda(l) \cdot f(\vec{W}, l) dl \tag{3}$$

is minimized, where $\lambda(l)$ is the distribution function of wirelength l, l_{min} and l_{max} are the minimum and maximum wirelengths for this metal layer, and $f(\vec{W}, l)$ is the objective function to be minimized by the design. In this study we choose $f(l)$ to be of the form $A^j(\vec{W}, l) \cdot T^k(\vec{W}, l)$, where $A(\vec{W}, l)$ and $T(\vec{W}, l)$ denote the area and delay using $\vec{W}$. For one-width design, $\vec{W}$ has only one component W. For two-width design, $\vec{W}$ has two components W_1 and W_2. If we set $j = 0$ and $k = 1$ in Eqn. (3), the objective is to

achieve the best delay. Insisting the minimum delay in wire sizing can be very costly in terms of wire width, as the delay/width curve is very flat while approaching the optimal delay. Our empirical study suggests that the AT^4 metric (i.e., $j = 1$ and $k = 4$) leads to area-efficient performance optimization in general. For example, it was shown in [21] that under the 0.10 μm technology, optimal one-width solution for a 2cm interconnect under the AT^4 metric uses over 60% smaller wiring area with only a 10% increase in delay compared to that obtained for delay optimization only. The wire width planning results presented in this section uses the AT^4 metric. But our solution technique is general for optimizing other metrics as well.

Our approach to the wire width planning problem is fairly straightforward. We find the best one-width or two-width pair to minimize the objective function in Eqn. (3) by exhaustive enumeration through all possible widths or all possible wire-width pairs, respectively. This method clearly cannot scale to find a wire-width planning solution with many widths. But this is not a problem, as we are only interested in finding a very small number of widths per layer as the planning solution. Using this approach, we in fact have achieved a rather surprising result which suggests that two *predetermined* wire widths per metal layer are sufficient to achieve near-optimal performance for *a wide range of nets* in that layer.

For example, for layers 7 and 8 in the $0.10\mu m$ technology, assuming that the wirelength in this layer pair is evenly distributed from 7.57mm to 24.9mm (according to the wirelength distribution model described in [21]), our wire-width planning tool suggests that the best one-width is $1.98\mu m$, and that the best two-width design consists of wires of widths $1.0\mu m$ and $2.0\mu m$. Table 3 shows the comparison of using the one-width, two-width, and many-width designs by running GISS (global interconnect sizing and spacing) algorithm presented in [22]. Three different pitch-spacings (denoted as pitch-sp in Table 3) between adjacent wires in layers 7 and 8 of the $0.10\mu m$ technology are used. For each pitch-sp, we compare the average delay, the maximum delay difference (in percentage) from GISS (ΔT_{max}) for all lengths, and the average width. For pitch-spacing of 2.0 μm, one-width design has an average delay about 14% and 20% larger than those from the two-width design and the many-width design, respectively. Moreover, it has an average wire width (thus area) about 1.83× and 1.92× of those from two-width design and many-width design, respectively. The two-width design, however, achieves close to the optimal delay as computed by the many-width design obtained by the GISS algorithm (just 3 to 5% larger) and uses only a slightly larger area (less than 5%) than that of the multi-width design

using GISS. When the pitch-spacing becomes larger, the differences between one-width, two-width, and many-width designs get smaller.

In Table 3, we also list the maximum delay difference (ΔT_{max}) between the one-width and two-width designs compared to the many-width design. It is an important metric as it can bound the error of the corresponding wire-width planning solution under *any length distribution function* $\lambda(l)$ in Eqn. (3) according to the following result.

Theorem 4.1 *If* $\left|\frac{f(\vec{W},l)-f(\vec{W}^*,l)}{f(\vec{W}^*,l)}\right| \leq \delta_{max}$ *for any* $l \in (l_{min}, l_{max})$, *then for any distribution function* $\lambda(l)$, *we have*

$$\left|\frac{\Phi(\vec{W}, l_{min}, l_{max}) - \Phi(\vec{W}^*, l_{min}, l_{max})}{\Phi(\vec{W}^*, l_{min}, l_{max})}\right| \leq \delta_{max}. \tag{4}$$

□

For the two-width design shown in the table (derived from uniform distribution $\lambda(l) \equiv 1$), since the maximum delay difference ΔT_{max} is only 3.9% to 7%, according to Theorem 4.1, one can conclude that this two-width design will differ from the optimal design (using possibly many widths) by at most 3.9% to 7% for *any distribution function* $\lambda(l)$. The reader may refer to [21] for more details.

Scheme	pitch-sp=2.0 μm			pitch-sp=2.9 μm			pitch-sp=3.8 μm		
	T_{avg}	ΔT_{max}	avg-w	T_{avg}	ΔT_{max}	avg-w	T_{avg}	ΔT_{max}	avg-w
one-width	0.245	28.2%	1.98	0.177	15.7%	1.83	0.143	5.9%	1.63
two-width	0.215	7.0%	1.08	0.167	5.9%	1.23	0.140	3.9%	1.41
many-width	0.204	-	1.03	0.159	-	1.19	0.136	-	1.38

Table 3: Comparison of using one-width design, two-width design and many-width design (up to 50× min width) using GISS for wire sizing and spacing. Layers 7 and 8 of $0.10\mu m$ technology are used, with wirelength ranging from 8.04 to $22.8mm$. Driver size is assumed to be 250× min size.

The fact that two widths are sufficient for each layer greatly simplifies the detailed routing problem (a full-blown gridless router may not be necessary) and possibly other problems, such as RC extraction and layout verification.

5 Interconnect Planning in an Interconnect-Centric Design Flow

In the past several years, our research group at UCLA has been developing a novel interconnect-centric design flow and methodology that emphasizes interconnect planning and optimization throughout the entire design process. Such a flow goes through the following three major design phases: (1) interconnect planning, which includes physical hierarchy generation, floorplanning with interconnect planning, and interconnect architecture planning; (2) interconnect synthesis, which determines the optimal or near-optimal interconnect topology, wire ordering, buffer locations and sizes, wire width and spacing, etc., to meet the performance and signal reliability requirements of all nets under the area and routability constraints; and (3) interconnect layout, which carries out detailed routing to implement the complex width and spacing requirements of all wires using a flexible and efficient multi-layer general-area gridless routing system.

Figure 8 shows an overview of our proposed interconnect-centric design flow. The two modules on the right-hand side are two supporting modules. The TRIO (Tree, Repeater, and Interconnect Optimization) library consists of a set of efficient interconnect optimization routines determines the optimal interconnect structure of each net in terms of interconnect topology, wire width and spacing, buffer locations and sizes, etc., to meet the performance and signal reliability requirements. The IPEM library consists of a set of fast and accurate *interconnect performance estimation models* (IPEMs) with consideration of various optimization techniques, including optimal wire sizing (OWS), simultaneous driver and wire sizing (SDWS), and simultaneous buffer insertion, buffer sizing and wire sizing (BISWS). These IPEMs are very efficient (constant run time in practice), and provide high-level abstraction. In addition, they provide explicit relations between the interconnect performance and layout design parameters under various kinds of optimization, which helps to make design decisions at high levels. These models have been tested on a wide range of parameters and have about 90% accuracy on average compared with those running complex optimization algorithms in TRIO directly (in terms of the delay measured by HSPICE simulations) [23, 21].

The modules in the center column of Figure 8 show the major steps in our interconnect-centric design flow: In order to cope with the design complexity of giga-scale integration in the nanometer technologies, we would

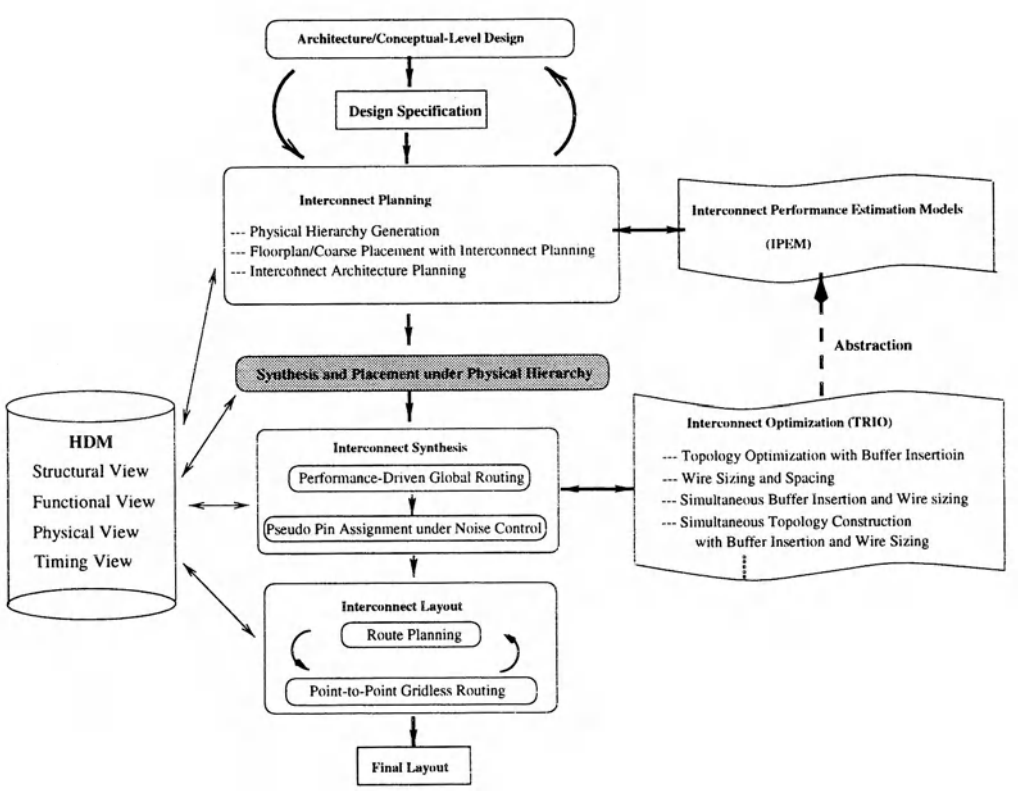


Figure 8: Overview of our interconnect-centric IC design flow.

like the designer (or the design team) focus on designs primarily at the architecture or conceptual level. Given a design specification (usually in a HDL specification such as Verilog or VHDL) as the output of the architecture or conceptual level design, our interconnect-centric flow first goes through the interconnect planning phase which transforms the functional hierarchy embedded in the HDL specification into a good physical hierarchy, performs coarse placement with global interconnect planning and interconnect architecture planning (when appropriate). It is possible for physical hierarchy generation to be performed together with coarse placement and global interconnect planning at the same time, as the global placement and interconnect planning usually influence the physical hierarchy generation. Interconnect performance estimation models (IPEMs) are used extensively during interconnect planning for predicting the performance of the optimized interconnects. After physical hierarchy generation, coarse placement with global interconnect planning, we shall have a good first-order estimation of the overall circuit performance (which is determined primarily by global interconnects). We can quickly provide feedback to the designer to indicate if the proposed architectural or conceptual level design is feasible. Therefore, the designer can quickly iterate with the interconnect planning tool to evaluate multiple architecture or micro-architecture designs, and converge to the most promising one(s) for further refinement.

After interconnect planning, the next of phase of the design flow is synthesis and placement for each module under the physical hierarchy, as shown in the shaded box in Figure 8. Currently, we are using off-the-shelf synthesis and placement techniques for this step (such as using the Design Compiler from Synopsys or the SIS/VIS package from UC Berkeley for logic synthesis and the TimberWolf or GordianL package for placement). We tend to believe once the physical hierarchy and global interconnects are defined, existing synthesis and placement algorithms can work well at the module level which contains mainly local interconnects, as argued in [24]. In particular, gain-based synthesis can be used to synthesize small to medium size logic blocks under the physical hierarchy [25, 26]. We are also starting a new project at UCLA on placement-driven synthesis to investigate if one can improve the result from this step significantly by combining synthesis and placement at the module level.

Once synthesis and placement for each module is determined, we perform interconnect synthesis, which includes performance-driven global routing with various interconnect optimizations for delay optimization, followed by pseudo pin assignment with noise optimization. The TRIO library is

used extensively during interconnect synthesis for determining the best interconnect structures for delay and noise optimization.

Finally, a gridless routing system is used to complete interconnect layout to implement various kinds of optimized interconnect structures. It includes a coarse grid based route planning engine and an efficient point-to-point gridless routing engine working on the implicit representation of the underlying non-uniform grid graph.

All these modules have been implemented, and they interact through a common hierarchical data model (HDM) shown on the left-hand side of Figure 8. The HDM provides a complete functional and physical representation of the design, including the structural view, the functional view, the physical view, and the timing view so that logic transformation, interconnect planning/optimization, or layout design can be carried out at every phase of the design process.

More detailed description of various modules in the flow is available from [27]. Currently, each module in this design flow has been fully verified and has shown very promising results. We are in the process of performing integrated test of the overall flow and design methodology. We are integrating all the modules into the proposed interconnect-centric design flow and running several complete designs through such flow. We hope to report complete experimental results in the near future. We believe that such an interconnect-centric design flow will effectively bridge the gap between high-level design abstraction and physical-level implementation, reduce or eliminate the uncertainty due to interconnects on system performance and reliability, and assure design convergence between synthesis and layout.

6 Conclusions

In this chapter, we first identified the needs for interconnect planning, which is a step missing in the current design flow, but the centerpiece of a interconnect-centric design flow for nanometer technologies. We divided the interconnect planning process into three steps: physical hierarchy generation, floorplanning/coarse placement with interconnect planning, and interconnect architecture planning. We developed efficient algorithms for each step and demonstrated the potential gains that one may achieve from interconnect planning. Finally, we showed how interconnect planning can be integrated in an interconnect-centric design flow. We would like to emphasize that although significant progress has been made on interconnect planning as

reported in this chapter, we are still actively working in this area to gain deeper understanding and search for better solutions.

Acknowledgments

The author would like to thank the students and researchers (current and former) at the UCLA VLSI CAD Laboratory who have contributed to the results on interconnect planning presented in this chapter. These include Chin-Chih Chang, Tianming Kong, Hon-Ching Peter Li, Sungkyu Lim, David Z. Pan, Chang Wu, and Xin Yuan. The author would also like to thank Tony Drumm from IBM for stimulating discussions regarding hierarchical designs and for providing the design plot shown in Figure 3. The author is very grateful to Xin Yuan for her assistance in preparing this manuscript. The research reported in this chapter is partially by Semiconductor Research Corporation under Contract 98-DJ-605, National Science Foundation Young Investigator Award MIP-9357582, MARCO/Gigascale Research Center, and a grant from Intel Corporation.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, vol. 38, pp. 114–117, April 1965.

[2] Semiconductor Industry Association, *National Technology Roadmap for Semiconductors*, 1997.

[3] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 1999.

[4] J. Cong, "Challenges and opportunities for design innovations in nanometer technologies," in *SRC Working Papers, http://www.src.org/prg_mgmt/frontier.dgw*, Dec. 1997.

[5] J. Cong, L. He, C.-K. Koh, D. Z. Pan, and X. Yuan, *UCLA Tree-Repeater-Interconnect-Optimization Package (TRIO)*, 1999, Available at http://cadlab.cs.ucla.edu/software_release/trio/htdocs/.

[6] P. Pan, A. K. Karandikar, and C. L. Liu, "Optimal clock period clustering for sequential circuits with retiming," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 489–498, 1998.

[7] J. Cong, H. Li, and C. Wu, "Simultaneous circuit partitioning/clustering with retiming for performance optimization," in *Proc. Design Automation Conf.*, pp. 460–465, 1999.

[8] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning : Application in VLSI domain," in *Proc. Design Automation Conf.*,.pp. 526–529, 1997.

[9] J. Cong and S. K. Lim, "Edge separability based circuit clustering with application to circuit partitioning," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 429–434, 2000.

[10] J. Cong and S. K. Lim, "Multiway partitioning with pairwise movement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 512–516, 1998.

[11] A. Brandt, "Multi-level adaptive solution to boundary value problems," *Mathematics of Computation*, vol. 31, no. 138, pp. 333–390, 1977.

[12] W. Briggs, *A Multigrid Tutorial.* SIAM, 1987.

[13] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Applications in vlsi domain," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 7, pp. 69–79, Mar 1999.

[14] T. Chan, J. Cong, T. Kong, and J. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *Proc. IEEE International Conference on Computer Aided Design*, pp. 171–176, Nov 2000.

[15] J. Cong and S. K. Lim, "Performance driven multiway partitioning," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 441–446, 2000.

[16] J. Cong and S. K. Lim, "Physical planning with retiming," in *Proc. Int. Conf. on Computer Aided Design*, pp. 2–7, Nov. 2000.

[17] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning," in *Proc. Int. Conf. on Computer Aided Design*, pp. 358–363, 1999.

[18] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[19] Z. Pan, *Interconnect Synthesis and Planning for High-Performance IC Designs.* PhD thesis, University of California, Los Angeles, 2000.

[20] P. Sarkar, V. Sundararaman, and C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning," in *Proc. Int. Symp. on Physical Design*, 2000.

[21] J. Cong and D. Z. Pan, "Interconnect estimation and planning for deep submicron designs," in *Proc. Design Automation Conf.*, pp. 507–510, June, 1999.

[22] J. Cong, L. He, C.-K. Koh, and Z. Pan, "Global interconnect sizing and spacing with consideration of coupling capacitance," in *Proc. Int. Conf. on Computer Aided Design*, pp. 628–633, 1997.

[23] J. Cong and D. Z. Pan, "Interconnect delay estimation models for synthesis and design planning," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 97–100, Jan., 1999.

[24] D. Sylvester and K. Keutzer, "A global wiring paradigm for deep submicron design," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, pp. 242–252, February 2000.

[25] I. E. Sutherland, R. F. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits.* Academic Press, Inc., 1999.

[26] http://www.magma-da.com/c/@4sqgeyXVifzmg/Pages/Gain_based_overview.html.

[27] J. Cong, "An interconnect-centric design flow for nanometer technologies," *Proceedings of the IEEE*, vol. 89, no. 4, 2001.

Modern Standard-cell Placement Techniques

Xiaojian Yang, Elaheh Bozorgzadeh, Majid Sarrafzadeh
Computer Science Department
University of California, Los Angeles, CA 90095
E-mail: xjyang,elib,majid@cs.ucla.edu

Maogang Wang
Simplex Solutions, Inc.
521 Almanor Avenue, Sunnyvale, CA 94086
E-mail: mgwang@simplex.com

Contents

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 45–87.
© 2001 *Kluwer Academic Publishers.*

1 Introduction

Placement is a classical problem in VLSI physical design. A lot of effective placement tools have been proposed in the last twenty years [1, 2, 3, 4, 5]. Previous work on the placement problem falls into two classes: constructive and iterative. It is generally believed that iterative approaches can produce better results than constructive approaches but are slower. The intense competition is driving the electronic design automation tools to finish designs in the shortest amount of time. When the circuit size gets larger and larger, quality degradation is expected for the flat iterative approaches. The multi-level hierarchical technique is regarded indispensable for solving today's complex VLSI placement problem without sacrificing quality. Most recent placement tools [1, 3, 4, 5] employ hierarchical approaches, including top-down annealing approaches [3, 5] and recursive quadratic methods [1, 4].

Orthogonal to the hierarchical approach, partitioning is also considered very effective when dealing with large circuits. Partitioning approaches are often used together with the hierarchical approaches. These hierarchical partitioning approaches are based on repeated division of a given circuit into subcircuits to optimize a given partitioning objective. However, they were out-performed by iterative placement tools in small circuits. Up to now, TimberWolf [3], a simulated annealing (SA) based placement tool, has been the performance leader among university standard-cell placement tools ever since its first release in 1985.

In addition to minimizing wirelength, meeting timing specification is a more important problem in modern VLSI physical design. A number of of timing-driven placement approaches have been proposed. These approaches can be classified into two major categories: path-based algorithms and net-based algorithms. Path-based approaches analyze path delays explicitly and try to satisfy both timing requirements and physical requirements simultaneously [6, 7]. Net-based algorithms transform timing constraints into weights [8, 9, 10] or budgets [11, 12, 13, 14] assigned to nets and use these weights and budgets to guide the placement process. Path-based approaches are more accurate but computationally expensive since a set of path delays have to be maintained during the placement stage. On the contrary, net-based approaches are inherently fast but lack accuracy. Both types of timing-driven approaches need to be constructed based on a good wirelength-driven placement framework.

Recently, with the advances in multi-level circuit partitioning technique [15, 16], several academic placement tools have been designed and implemented. *Capo* [17], *Feng Shui* [18] and *Dragon* [19] are three examples.

They have the following common features:

- They can handle large industrial circuits (e.g. 200,000 modules);
- They employ the multi-level partitioning technique, *hMetis* [15] or *ML-Part* [16];
- They can produce good placements in terms of total bounding box wirelength within a short amount of time.

In this chapter we investigate one of these placement tools, named *Dragon*, to review the modern techniques in circuit placement. We argue that the top-down hierarchical approach is the correct way to solve the large sized placement problem. The study on the interaction between wirelength and net-cut objectives shows that minimizing net-cut is an important shortcut to reduce total wirelength in placement. We also propose a *"+1 level"* cluster based moving technique to improve wirelength objective, and examine the effect of traditional terminal propagation technique in different placement schemas. Experiments to evaluate the placement approaches are performed on both old MCNC benchmark suits and recent industrial circuits [1]. Comparing to highly optimized commercial placement tool, *iTools* (formerly TimberWolf), Dragon can produce slightly better placement results using much less amount of runtime (2× speedup) on a single processor machine.

We also discuss a priori wirelength estimation based on the well known Rent's Rule [21]. Two different Rent exponents, partitioning Rent exponent and placement Rent exponent, are defined in this chapter. The relationship between them is analyzed and experimentally validated. The experiments show that the Rent exponent obtained from placement output could be a measure of quality of the placement tool.

The rest of the chapter [2] is organized as follows: In Section 2, we briefly describe the flow and algorithms used in Dragon. In Section 3, we theoretically analyze the relationship between the net-cut and the wirelength objective and argue that net-cut is important in solving placement problem. In Section 4, we will explain the detailed implementation of Dragon.

[1] Almost all previously published placement algorithms use MCNC standard-cell benchmark suite for testing. This suite was released in 1992 with most circuits having less than 30,000 cells. However, circuits designed in today's industry have typically more than 100,000 cells. In 1998, Alpert modified and released 18 industrial circuits from IBM to form the ISPD98 partitioning benchmark suite [20]. They are much larger than MCNC circuits (from 10,000 to 200,000 cells). Although they were originally released for the purpose of partitioning, they can also be modified and used in placement.

[2] Part of work related to this chapter has been published in several conferences: [19] in *ICCAD 2000*, [22] in *ISPD 2000*, and [23] in *SLIP 2001*.

Comparison experimental results will be shown in Section 5. In section 6 we discuss the wirelength estimation problem based on the analysis of different Rent exponents. The conclusion remarks are made in Section 7.

2 Overview of Dragon

As circuits get larger, the placement problem for VLSI physical design can only be solved effectively using hierarchical approaches due to the exponentially growing solution space. VLSI designers tend to design circuits hierarchically. This also gives hierarchical placement algorithms big advantages over other placement algorithms. The top-down based hierarchical approach is the backbone of our placement tool, Dragon. In this section, we will have a brief overview of the general hierarchical placement approaches and algorithms used in Dragon.

A typical top-down hierarchical placement approach can be generalized as follows: at a given hierarchical level, the layout area is partitioned into several global bins. All cells of the circuit will be distributed into these global bins to minimize a certain placement objective. This cell distribution problem is called a hierarchical placement problem. If a cell is distributed into a particular global bin, it will be placed within the area of this bin in the final layout. As we proceed to finer levels, the number of global bins increases and the physical size of global bins decreases. Thus we can get more and more detailed information about physical locations of cells as we proceed. The top-down hierarchical approach will terminate when there are only a few cells in each global bin.

Dragon can be divided into two phases, global placement (GP) and detailed placement (DP). A top-down hierarchical approach is used in the GP phase. We recursively solve the hierarchical placement problem and quadrisect each global bin into four smaller bins at each level. Overlap between cells are allowed in the GP phase. In fact, all the cells belong to the same bin are placed at the center of the bin. The DP phase takes the output from GP and produces an overlap free layout. Then it iteratively improves the legal layout using a greedy heuristic. Due to the computational complexity, the DP heuristic is only capable of performing local optimization. Thus it is expected that the top-down hierarchical GP phase can do majority of work in placement.

Wirelength and net-cut are two popularly used objectives in different hierarchical placement algorithms. It is commonly believed that partitioning tools (minimizing net-cut) are much more mature and effective than

wirelength minimization tools. On the other hand, wirelength at different hierarchical levels is a more accurate estimation of the final wirelength than net-cut. In order to achieve high performance, we integrate wirelength and net-cut together in the GP phase of Dragon to take advantage of both objectives. Although these two objectives are widely used in other placement tools, the relationship between them are not very well understood. In the next section, we will analytically study the properties of them and the reason why we can use net-cut minimization methods in placement. The detailed implementation of Dragon will be explained in Section 4.

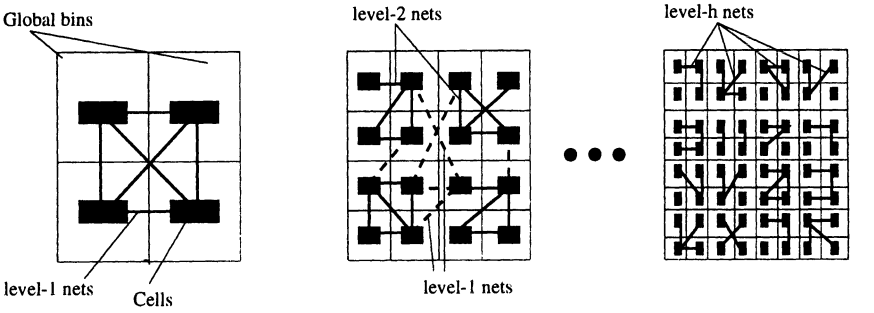
3 Net-cut as a Good Wirelength Objective

To simplify the problem a bit, we study the quadrisection method in the top-down context. We assume the first hierarchical level contains four global bins and we have H hierarchical levels in total. Thus there are 4^h global bins at a generic level h and 4^H global bins at the final level. Cells belonging to any global bin are placed at the center of that bin. As a consequence, the length of any net is multiples of the width and height of global bins. We can therefore normalize the wirelength at each level using the width and height of global bins. If all cells connected to a net are located in the same global bin, the length of the net is zero [3] and the net is not cut. Otherwise, the net is cut. More net are being cut when the top-down hierarchical approach goes to lower levels. We call those nets that are cut at level h and not being cut at any previous levels "level-h cut-nets" or "level-h nets" for simplicity. The number of level-h nets is called level-h net-cut and denoted by C_h (shown in Figure 1). Therefore, the total number of net-cut at level h would be $\sum_{i=1}^{h} C_i$.

Typically, the aspect ratio of the layout is close to 1, thus global bins at each level have similar width and height [4]. If we use the width and height of global bins at the final level (level H) as the length unit, the width and height of global bins at level h is $2^{(H-h)}$. If a net is a level-h net, by definition, the net is not cut until level h. Thus all the cells of this net should be inside a global bin in level $(h-1)$ which has a width and height of $2^{(H-h+1)}$. As illustrated in Figure 2, in the worst case, the length of the net at the final level H is $2 \cdot 2^{(H-h+1)}$. We can also use statistical knowledge to get a better prediction of level-h net length. Caldwell, etc presented a nice theoretical work in predicting wirelength for different nets in [24]. However,

[3] Generalization to non-zero length for such nets is straight forward.

[4] It is very easy to extend our results to the case when aspect ratio is not equal to 1.



(a) Level 1 has 4 global bins the number of level-1 net-cuts is C_1

(b) Level 2 has 16 global bins the number of level-2 net-cuts is C_2

(c) Level h has 4^h global bins the number of level-h net-cuts is C_h

Figure 1: Net-cuts at different levels in a top-down hierarchical approach.

the results shown in [24] are too complicated. We can get a much more simple prediction if we assume all nets have only two terminals. Under the assumption of two-terminal nets and uniform distribution of terminals, the statistical expected length of a level-h net is $\frac{7}{9} \cdot 2^{(H-h+1)}$. We expect a slightly longer length for multi-terminal nets. In practice, since cells are placed to minimize wirelength, the actual length of a level-h net could be even less than the statistical length, $\frac{7}{9} \cdot 2^{(H-h+1)}$. Let us denote the average length of level-h nets by a real number l_h. Since the number of level-h nets is C_h, we can express the total wirelength for level-h nets in the worst, statistical and the average case as $C_h \cdot 2 \cdot 2^{(H-h+1)}$, $C_h \cdot \frac{7}{9} \cdot 2^{(H-h+1)}$ and $C_h \cdot l_h$, respectively. Therefore, in a top-down hierarchical approach, the total wirelength in these three cases can be expressed as:

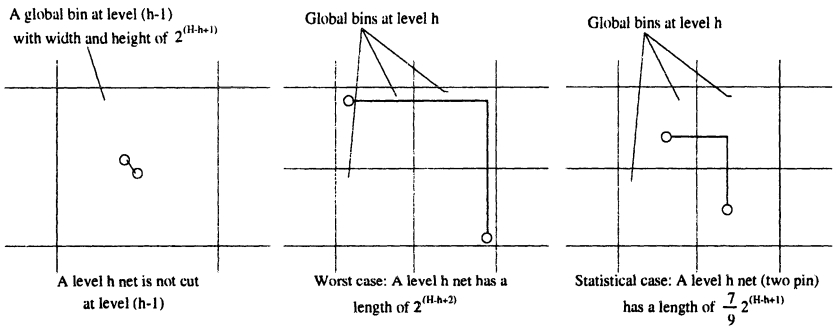


Figure 2: Wirelength of a level-h net in the worst and statistical case.

$$WL_{worst} = \sum_{i=1}^{H} C_i \cdot 2 \cdot 2^{(H+1-i)} \tag{1}$$

$$WL_{stat} = \sum_{i=1}^{H} C_i \cdot \frac{7}{9} \cdot 2^{(H+1-i)} \tag{2}$$

$$WL_{ave} = \sum_{i=1}^{H} C_i \cdot l_i \tag{3}$$

Next, we will use Rent's rule to show the relationship between C_i and C_{i-1}. Rent's rule was first described by Landman and Russo in 1971 [21]. When we partition a circuit into several blocks, Rent's rule tells the relationship between the number of external pins on each block and the number of cells inside each block. Let us denote the average number of cells in a block by B_m, then the average number of external pins P_m can be expressed as $P_m = T_b B_m^r$ where $0 \leq r \leq 1$ is called the Rent parameter and T_b is the average number of terminals per block. Rent's rule is experimentally validated for a lot of real circuits and for different partitioning methodologies. For real circuits, the Rent parameter r usually has a value between 0.3 and 0.8.

Theorem 3.1 *If a circuit obeys Rent's rule and has a Rent parameter of* r*, in a top-down quadrisectional approach, the number of level-i nets* C_i *and the number of level-(i-1) nets* C_{i-1} *has the relationship of* $\frac{C_i}{C_{i-1}} = 4^{1-r}$

Proof:

At any hierarchical level h in a top-down quadrisectional approach, the number of global bins is 4^h, thus we have $\frac{N_c}{4^h}$ cells in each global bin where N_c is the number of total cells in the circuit. According to Rent's rule, the number of external pins on each global bin $P_m = T_b(\frac{N_c}{4^h})^r$. The total number of external nets will be

$$Cut_h = \frac{P_m}{p_{avg}} 4^h = \frac{T_b}{p_{avg}} \cdot (\frac{N_c}{4^h})^r \cdot 4^h \tag{4}$$

p_{avg} is the average number of terminals per net. When the number of global bins is equal to the number of total cells N_c, all the nets are cut.

Thus we have the relationship of $N_n = \frac{T_b}{p_{avg}} \cdot N_c$ where N_n is the number of total nets in the circuit. Therefore, we can rewrite (4) using N_c and N_n as follows:

$$Cut_h = \frac{N_n}{N_c} \cdot (\frac{N_c}{4^h})^r \cdot 4^h$$

By definition, level h net-cut $C_h = Cut_h - Cut_{h-1}$. Thus we have:

$$C_h = Cut_h - Cut_{h-1} = \frac{N_n}{N_c} \cdot (\frac{N_c}{4^h})^r \cdot 4^h \cdot (1 - 4^{r-1})$$

and the ratio between C_i and C_{i-1} is:

$$\frac{C_i}{C_{i-1}} = \frac{\frac{N_n}{N_c} \cdot (\frac{N_c}{4^i})^r \cdot 4^i \cdot (1 - 4^{r-1})}{\frac{N_n}{N_c} \cdot (\frac{N_c}{4^{i-1}})^r \cdot 4^{i-1} \cdot (1 - 4^{r-1})} = 4^{1-r} \quad \blacksquare$$

Theorem 3.1 tells us that the ratio between C_i and C_{i-1} is a constant. For industrial circuits, r has an value between 0.3 and 0.8, thus the ratio has a value between 1.3 and 2.6.

With the help of Theorem 3.1, we can get the total wirelength at the final level H for the worst and the statistical case.

Lemma 3.2 *In a top-down quadrisectional approach, the total wirelength at the final level H is $C_1 \cdot 2^{H+1} \cdot \frac{1-(\frac{\alpha}{2})^H}{1-\frac{\alpha}{2}}$ in the worst case, where α is the ratio between C_i and C_{i-1} ($\alpha = 4^{1-r}$ from Theorem 3.1).*

Proof:

Since $C_i = \alpha C_{i-1}$, $C_i = \alpha^{i-1} \cdot C_1$. Equation (1) becomes:

$$WL_{worst} = \sum_{i=1}^{H} \alpha^{i-1} \cdot C_1 \cdot 2 \cdot 2^{(H+1-i)} = C_1 \cdot 2^{H+1} \cdot \frac{1 - (\frac{\alpha}{2})^H}{1 - \frac{\alpha}{2}} \quad \blacksquare$$

Lemma 3.3 *In a top-down quadrisectional approach, the total wirelength at the final level H is $\frac{7}{9} C_1 \cdot 2^H \cdot \frac{1-(\frac{\alpha}{2})^H}{1-\frac{\alpha}{2}}$ in the statistical case, where α is the ratio between C_i and C_{i-1}.*

Proof:
Since $C_i = \alpha C_{i-1}$, $C_i = \alpha^{i-1} \cdot C_1$. Equation (2) becomes:

$$WL_{stat} = \sum_{i=1}^{H} \alpha^{i-1} \cdot C_1 \cdot \frac{7}{9} \cdot 2^{(H+1-i)} = \frac{7}{9} C_1 \cdot 2^H \cdot \frac{1-(\frac{\alpha}{2})^H}{1-\frac{\alpha}{2}} \quad \blacksquare$$

Equation (3) cannot be further simplified without some knowledge of l_i's. Intuitively, the average wirelength for lower level net should be larger than the average wirelength for higher level nets. However, the values of l_i's are very hard to model theoretically. We use empirical data to model l_i's instead. A top-down quadrisectional approach is used to recursively place cells at each level. A nets is labeled as a level-i net if it is first cut at level i. We record the average length for level-i nets ($i = 1, 2, ..., H$) at the final level H. The length unit is set to be the width/height of the global bins at the final level H. Table 1 shows the experimental values of l_i's for a number of benchmark circuits with $H = 6$. As we can see, l_i values at different levels have a similar relationship as C_i values which is expressed in Theorem 3.1. We can model it as $l_i = \lambda l_{i+1}$ where λ is approximately a constant. We can use this empirical model to obtain the total wirelength at the final level H in the average case.

Ckts	l_1	l_2	l_3	l_4	l_5	l_6
ibm01	27.8	19.1	10.4	5.8	3.3	1.7
ibm02	28.3	21.9	12.6	6.6	3.6	1.9
ibm03	38.4	18.9	10.9	5.8	3.4	1.8
ibm04	32.8	18.1	10.2	5.9	3.5	1.9
ibm05	41.9	27.1	13.0	7.7	4.3	2.7

Table 1: Value of l_i at different levels.

Lemma 3.4 *In a top-down quadrisectional approach, the total wirelength at the final level H is* $C_1 l_1 \cdot \frac{1-(\frac{\alpha}{\lambda})^H}{1-\frac{\alpha}{\lambda}}$ *in the average case, where* α *is the ratio between* C_i *and* C_{i-1} *and* λ *is the ratio between* l_i *and* l_{i+1}.

Proof:
Since $l_i = \lambda l_{i+1}$, $l_i = (\frac{1}{\lambda})^{i-1} \cdot l_1$. We also have $C_i = \alpha^{i-1} \cdot C_1$. Equation (3) becomes:

$$WL_{ave} = \sum_{i=1}^{H} \alpha^{i-1} \cdot C_1 \cdot (\frac{1}{\lambda})^{i-1} \cdot l_1 = C_1 l_1 \cdot \frac{1-(\frac{\alpha}{\lambda})^H}{1-\frac{\alpha}{\lambda}} \quad \blacksquare$$

Lemma 3.5 *In a top-down quadrisectional approach, the total wirelength at the final level H is always greater than $C_1 \cdot \frac{1-\alpha^H}{1-\alpha}$, where α is the ratio between C_i and C_{i-1}.*

Proof:

At the final level H, each net-cut contributes at least one unit to the wirelength. Therefore, the lower bound for the total wirelength at the final level H is

$$WL \geq \sum_{i=1}^{H} C_i = \sum_{i=1}^{H} \alpha^{i-1} \cdot C_1 = C_1 \cdot \frac{1-\alpha^H}{1-\alpha} \quad \blacksquare$$

Lemma 1, 2, 3, 4 express the total wirelength at the final level H using circuit parameter α and λ. α and λ have similar values in different industrial circuits. For instance, according to Theorem 3.1, the value of α is ranging from 1.3 to 2.6. Table 1 shows that λ also has a value around 2. In order to get a more realistic feeling of Lemma 1, 2, 3, 4, we let both α and λ be a real number of 2. Judged by the Table 1, the value of l_1 is comparable to a half of the width/height of the layout ($l_1 \sim 2^{H-1}$).

When $\alpha = 2$ and $\lambda = 2$, according to Lemma 1, 2, 3, 4

$$WL_{worst} = C_1 \cdot 2^{H+1} \cdot H \tag{5}$$

$$WL_{stat} = \frac{7}{9} C_1 \cdot 2^H \cdot H \tag{6}$$

$$WL_{ave} = C_1 l_1 \cdot H \sim C_1 \cdot 2^{H-1} \cdot H \tag{7}$$

$$WL \geq C_1 \cdot (2^H - 1) \tag{8}$$

Comparing Equation (5), (6), (7) and (8), the total wirelength in the worst, statistical and average case are all approximately H times larger than the lower bound of the total wirelength. Since H is the final hierarchical level where each global bin contains more than one cell, $H \leq \log N_c$.

Theorem 3.6 *In a top-down quadrisectional approach, the total wirelength at the final level H is between the total net-cut and the total net-cut times $2H$: $Cut \leq WL \leq (2 \log N_c) \cdot Cut$*

Theorem 3.2 can be easily proved by combining Equation (5) and (8). It suggests that the wirelength result produced by a top-down hierarchical approach is bounded by a factor of $O(\log N_c)$. Cut and wirelength in a top-down hierarchical approach are indeed correlated. In Section 4.1, experimental results will also show that net-cut minimization in the hierarchical approach is crucial to get a good final placement result.

4 Detailed Implementation of Dragon

As earlier mentioned in Section 2, Dragon consists of two phases: the GP phase and the DP phase. The top-down hierarchical approach is used in the GP phase. We integrate net-cut and wirelength together to solve the hierarchical placement problem at each level. Specifically, we start our GP from level 1 with four global bins. We go from level h to level $h+1$ by partitioning each subcircuit in a global bin at level h into four parts to reduce net-cut. Global bins at level h will be split into four smaller bins correspondingly. Thus there will be 4^{h+1} global bins and 4^{h+1} subcircuits at level $h+1$. Finally, we switch all 4^{h+1} subcircuits in level $h+1$ locally to minimize wirelength. GP terminates when each global bin contains less than about seven cells.

A traditional top-down hierarchical placement approach confines locations of subcircuits at level $h+1$ within the region of the global bin where the subcircuits reside in at the previous level h. This approach can greatly reduce the computational complexity. However, it can never correct wrong decisions made at higher levels. Our GP does not confine locations of subcircuits. This gives cells more freedom to move to achieve better placement results at each level. In order to reduce runtime, we limit our wirelength optimization searches in a local range.

We pick hMetis [15, 25] as the partitioner for Dragon because of its superior quality and friendly user interface. Simulated annealing is picked as the base algorithm to minimize wirelength because it is very easy to implement. A low temperature simulated annealing algorithm is also very effective in doing local optimization and reasonably fast. Since we are expecting a high quality output from GP, relatively little work need to be done in DP. The DP phase of Dragon uses a greedy algorithm to perform local optimiza-

tion and improve the quality of final placement iteratively. We will discuss implementation details of Dragon in the following subsections.

4.1 Interactions between Wirelength and Net-cut

As argued in Section 3, circuit partitioning is a nice shortcut to get a good wirelength minimized placement at each level. At each level in GP, we quadrisect each subcircuit inside a global bin into four smaller subcircuits. Since no physical location information is considered in partitioning, we then switch subcircuits in all global bins to reduce wirelength. However, net-cut is generally considered less accurate than wirelength. Several variations can be used in order to fix this fact. Based on previous work and intuition, we tried four approaches. Figure 3 illustrates these approaches. Assume we are about to partition the subcircuit within global bin B_0. Cell 1, 2, 3, 4, 5, 6, 7 are inside B_0. Cell 8, 9, 10, 11 are outside B_0 but have connections to cells inside B_0. We denote an n-terminal net by net $(c_1, c_2, ..., c_n)$, where c_1, c_2, ..., c_n are terminal cells of the net.

1. Approach A (Figure 3a): When we partition a subcircuit in a global bin at any level, nets which have terminal cells outside this global bin will be ignored. We do this because this is always cut no matter how we distribute those inside cells. In Figure 3a, net $(2, 6, 11)$, $(5, 6, 10)$, $(4, 7, 8, 9)$, $(7, 9)$ are ignored when partitioning subcircuit in B_0.

2. Approach B(Figure 3b): When we partition a subcircuit in a global bin at any level, terminal cells which are outside this global bin are ignored. In Figure 3b, since terminal cell 8, 9, 10, 11 are ignored, original net $(2, 6, 11)$, $(5, 6, 8)$, $(4, 7, 8, 9)$, $(7, 9)$ become new net $(2, 6)$, $(5, 6)$, $(4, 7)$, respectively. This method will encourage us to group the remaining inside terminal cells together even there are always outside terminal cells.

3. Approach C(Figure 3c): In approach A and B, we isolated the subcircuits within a global bin by removing connections between inside cells and outside cells. Intuitively, this is not good. The idea of "terminal propagation" was proposed in [26] to solve this problem. It adds to the current subcircuit *dummy cells* that are fixed in the appropriate partitions. In Figure 3c, cell 8 is mapped to a fixed vertex in the lower-left part of B_0; cell 9 and 10 are mapped to two fixed vertices in the lower-right part of B_0; cell 11 is mapped to a fixed vertex in the

upper-right part of B_0. This approach encourages cells be distributed in a global bin close to their outside neighbor cells.

4. Approach D(Figure 3d): In all above approaches, subcircuits are switched to reduce wirelength after partitioning is done. Instead of moving all the cells in each subcircuit together, we can also switch single cells around to reduce wirelength at this level. This idea was first proposed in [5]. Wirelength obtained by this approach should be better than wirelength obtained by other three approaches. However, it is not clear whether a optimal wirelength placement at any hierarchical level will produce a good final placement.

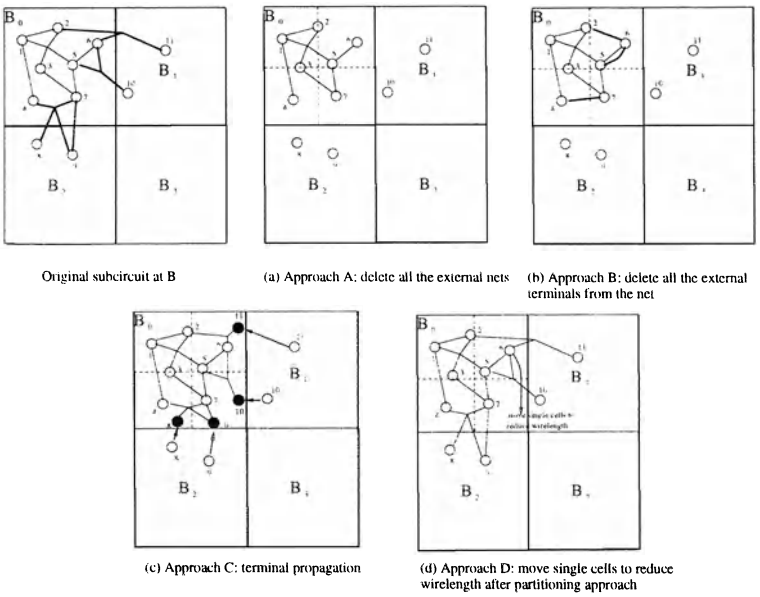


Figure 3: Graphical illustration of Approach A, B, C and D.

In order to find out which approach performs the best, we test all four approaches on several benchmark circuits. Table 2 shows the experimental results. The best result for each circuit among all four approaches is shown in the bold face. Quite surprisingly, approach B out-performs other approaches including approach C (terminal propagation) which is widely accepted and used in other placement tools.

In placement tools which use terminal propagation, subcircuits at each level are confined within the region of the global bin where they belong to at previous levels. We do not confine locations of subcircuits at each level

because we perform the post bin swapping stage to reduce the overall wirelength after the subcircuits are formed. This post bin swapping stage is used in approach A, B and C. The use of the post bin swapping stage might be a reason why approach B works better in our GP than approach C does. To further investigate this interesting issue, we implemented the conventional min-cut scheme with and without terminal propagation. The conventional scheme with terminal propagation is basically the same as approach C except it does not perform post bin swapping. Similarly, the scheme without terminal propagation is the same as approach B without the post bin swapping stage. Table 3 shows the experimental results comparing two conventional min-cut schemes and their counter parts in our approaches (approach B and C). Indeed, the terminal propagation can improve the wirelength results over the non terminal propagation min-cut scheme. However, the post bin swapping stage can help improve performance: approach C (terminal propagation + post bin swapping) outperforms pure terminal propagation. Finally, it is very interesting to find that the widely used terminal propagation scheme actually degrade performance while the post bin swapping stage is used (approach B is better than approach C).

Another interesting fact is that approach D is not successful either. This fact suggests that conserving connecting information between cells is more important than greedily obtain a wirelength optimal placement at each level. Minimizing net-cut not only can obtain placement results fast at each level, it also helps to improve the final placement quality.

Ckts	#cells	App. A	**App. B**	App. C	App. D
ibm01	12282	4.79	**4.71**	4.98	4.81
ibm02	19321	**13.70**	13.91	14.38	13.99
ibm03	22207	13.12	**12.83**	13.02	12.93
ibm04	26633	17.66	**16.58**	17.54	17.21
ibm05	29347	38.94	**38.21**	39.32	39.12

Table 2: Comparison of four different approaches.

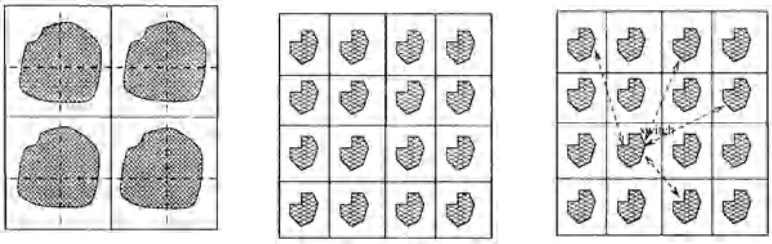
4.2 Optimizing Wirelength at a Hierarchical Level

Based on the analysis in Section 3, we believe a combination between the net-cut and the wirelength objective will be very effective in the hierarchical placement (as is currently done in several commercial packages). At a certain hierarchical level, we can first optimize net-cut. Then we further reduce

Ckts	mincut w/o term. prop.	mincut w/ term. prop.	App. C	**App. B**
ibm01	6.05	5.36	4.98	**4.71**
ibm02	16.77	15.04	14.38	**13.91**
ibm03	17.29	14.05	13.02	**12.83**
ibm04	22.58	18.34	17.54	**16.58**
ibm05	52.35	49.09	39.32	**38.21**

Table 3: Comparison of conventional min-cut schemes and our approaches.

wirelength based on the net-cut optimized placement. We can achieve this by moving cells around the global bins to optimized the wirelength. However, it would be very slow and ineffective if we only move a single cell at a time. We propose a "+1 *level clustering*" technique to perform this task effectively: Given a hierarchical level h which has N_b global bins, first we solve the net-cut optimization problem at hierarchical level $h+1$ which has gN_b global bins where g is defined in Section 2 (usually $g = 2$ or 4). Based on the net-cut optimization result at level $h+1$, we have gN_b cell clusters with each cluster being the set of all the cells in one global bin at level $h+1$. Then we go back to the given hierarchical level h and do the wirelength optimization by placing these gN_b clusters into N_b global bins. Figure 4 shows an example of this +1 level clustering technique which $g = 4$ and $N_b = 4$. This technique will be much faster than the single cell moving algorithm because we only need to place gN_b objects into N_b places.



(a) At current level h, cut the subcircuit in each bin into four parts.

(b) Global bins are also split into four smaller bins. Each smaller subcircuit will be placed into one global bin (level h+1).

(c) do wirelength optimization optimization by moving subcircuits around.

Figure 4: The "+1 level" clustering technique to improve wirelength.

The +1 level clustering technique can also be expanded to be a +2

,+3 or $+\delta$ level clustering technique. As δ increases, the run time will increase accordingly. If there is only one cell at each global bin at level $h+\delta$ ($\delta = \frac{\log|\mathcal{C}|-\log N_b}{logg}$), the $+\delta$ level clustering technique reduces to the single cell moving strategy.

Assuming we want to get a hierarchical placement with an optimized wirelength at level h which has N_b global bins, a step by step procedure for the +1 level clustering technique can be written as:

1. Obtain a net-cut optimized placement at level $h+1$.
2. Cluster cells in the same global bin at level $h+1$ together. There are gN_b clusters in total.
3. Do cluster placement at level h to minimize wirelength using the clusters obtained in Step 2.

We can also have a +0 level clustering technique which is to do the clustering and the wirelength optimization at the same hierarchical level h. The hierarchical placement obtained from the +0 level clustering technique is just a net-cut optimized placement since it has the optimal net-cut cost. We call this +0 level clustering technique the flat clustering technique at level h since it does not go to level $h+1$.

In the +1 level clustering technique, we use hMetis [15] as the tool to get the net-cut optimized placement. Benchmark results showed that hMetis performs really well on large sized circuits. HMetis can also handle multiway partitioning easily. By using hMetis as our net-cut objective optimizer, we have three variations on the +1 level clustering technique. Assume that we are working on the hierarchical level h with N_b global bins:

1. +1 level A: We use hMetis to get the net-cut optimized cell clusters at level $h+1$. Then we perform the wirelength optimization at level h using simulated annealing (since the number of moving 'items' is small, a near optimal wirelength can be obtained by the annealing).
2. +1 level B: We use hMetis to get the net-cut optimized placement at level h. Then we use hMetis to partition the subcircuit in each global bin into g clusters. Then we perform the wirelength optimization at level h with these gN_b clusters using simulated annealing.
3. +1 level C: We use hMetis to get the net-cut optimized placement at the first hierarchical level h_1. Then we use hMetis to keep going down by splitting global bins until we reach level $h+1$. Then we do

clustering at level $h+1$ and perform the wirelength optimization back at level h using simulated annealing.

When we split global bins to get from the level h_i to the level h_{i+1}, we use hMetis to do a g-way partition on the subcircuit in each global bin at level h_i. The subcircuit contains all the cells and terminals in that global bin. Nets in the subcircuit are modified so that they only contain terminals located in that global bin.

It is interesting to notice that given long enough time, the results from +1 level B should be no worse than the results from the flat clustering approach at level h.

We conduct experiments to compare the wirelength results from a wirelength optimized approach, the flat clustering approach and the three +1 level clustering approaches (approach A, B and C) at different hierarchical levels. For consistency and simplicity, we use simulated annealing to implement the wirelength optimization in the hierarchical placement. This wirelength optimization algorithm is implemented without using any clustering technique. We tested four circuits on 2×2, 4×4, 8×8, 16×16 and 32×32 hierarchical levels Wirelength and runtime comparison for all circuits are shown in the Table 4 - 7. The unit for the runtime is second cpu time.

	2×2		4×4		8×8		16×16		32×32	
technique	WL	time	WL	time	WL	time	WL	time	WL	time
WL opt.	3912	5903	3479	5800	4044	5769	4220	6090	6463	5818
flat clus.	2116	1527	3507	1998	4424	2912	4670	4589	4803	6985
app. A	2281	1115	3986	1055	4326	1381	4657	1806	5006	2250
app. B	2116	1044	3513	953	4384	1235	4556	1542	4533	2010
app. C	2136	1075	3836	914	4547	997	4612	1149	6324	1645

Table 4: Wirelength and runtime comparison between different approaches for circuit ibm05

Experiments on all tested circuits show a similar trend: at coarser levels (lower than 8×8 for most circuits), the flat clustering approach is the best among all approaches. The wirelength minimization technique actually produces worse wirelength than the net-cut minimization technique in this case. This is because that minimizing net-cut is almost equal to minimizing wirelength at coarser levels and the partitioning tool we used is extremely effective. Therefore we can get a near optimal net-cut cost which is close to an optimal wirelength cost in this hierarchical level very fast. On the other

	2 × 2		4 × 4		8 × 8		16 × 16		32 × 32	
technique	WL	time	WL	time	WL	time	WL	time	WL	time
WL opt.	4835	4285	5393	9719	5422	6013	6246	4364	7849	4062
flat clus.	3176	1513	3854	1771	4483	2382	5482	3448	6277	4966
app. A	3175	1232	4146	980	4958	1118	5965	1298	6846	1821
app. B	3176	1215	4061	991	4536	1078	5413	1152	6133	1492
app. C	3203	1178	4552	1089	5127	990	5606	1123	7784	1443

Table 5: Wirelength and runtime comparison between different approaches for circuit ibm06

	2 × 2		4 × 4		8 × 8		16 × 16		32 × 32	
technique	WL	time	WL	time	WL	time	WL	time	WL	time
WL opt.	10123	17389	11994	24894	14700	8231	12025	20662	14454	17743
flat clus.	8335	3535	9359	2492	11005	3208	12256	4491	14620	5323
app. A	8871	2349	11272	1812	14476	1740	16760	2034	20612	2766
app. B	8414	2368	10335	1592	11612	1614	12422	1771	13713	2187
app. C	8407	2335	10070	1645	11594	1626	12287	1814	13616	2168

Table 6: Wirelength and runtime comparison between different approaches for circuit ibm09

	2 × 2		4 × 4		8 × 8		16 × 16		32 × 32	
technique	WL	time	WL	time	WL	time	WL	time	WL	time
WL opt.	55031	68560	59501	40593	64548	15471	61236	33458	65001	25111
flat clus.	54878	4055	59154	4429	62037	5651	65506	8988	68071	10700
app. A	57348	4049	62674	2916	66580	2987	70528	3662	73478	4806
app. B	56703	4024	60197	2923	61299	2667	62962	3088	64918	3924
app. C	56269	3965	60075	2818	61842	2708	62891	3033	64520	3585

Table 7: Wirelength and runtime comparison between different approaches for circuit ibm12

hand, since the wirelength optimization technique is not as effective as the partitioning technique, given a certain amount of time, the wirelength optimization technique fails to get a near optimal wirelength cost. Results from approach A, B and C are not better than the results from the flat clustering approach. This fact suggests that the wirelength cost obtained from the flat clustering approach is really a good approximation of the optimal value because further breaking clusters does not help reduce wirelength.

At finer hierarchical levels, the performance of the flat clustering approach becomes worse. The partitioning tool we used is still powerful. The problem is that an optimal net-cut placement is getting further away from the optimal wirelength placement at finer levels. Thus we see better results produced by the wirelength optimization technique than the flat clustering technique. Therefore, it is wise to use the net-cut objective at early hierarchical levels and start considering wirelength at later levels. This is consistent with the analysis we made in the previous section. At finer levels, the proposed +1 level clustering technique works better than the wirelength optimization technique based on single cell moving strategy. This fact is reflected on the runtime comparison. The +1 level clustering technique is much faster because it has much fewer moving "clusters". Since the +1 level clustering technique is faster, within a certain amount of time, it usually produces better results than the single cell wirelength optimization technique. Of the three approaches (A, B and C) for the +1 level clustering technique, experiments show that approach A is always the worst and approach B is the best on average.

4.3 The Final Stage of Global Placement

The subsection 4.1 shows that a minimum wirelength placement at each level does not help to produce a good final placement. However, we find that such a "single cell switching" strategy to minimize wirelength is extremely helpful in the last level of GP where there are about seven cells per global bin. After GP stops at the last level, we switch single cells locally to minimize wirelength. We use a low temperature simulated annealing algorithm in this final stage of GP. As shown in [5], since the number of possible locations for each cell is the number of global bins, the size of solution space is greatly reduced. Therefore, performing annealing at this stage is much less expensive than performing annealing in the DP phase. In fact, we pick simulated annealing as the wirelength minimization approach in GP is mainly due to the ease of implementation. Other fast local optimization approaches (e.g., branch-and-bound search) can also be used to further speed up GP. Table

8 shows the comparison of the final placement wirelength using this final stage vs. not using this stage.

Ckts	w/o final stage	w/ final stage	% impr.
ibm01	4.99	4.70	5.8%
ibm02	14.71	13.76	6.5%
ibm03	13.56	12.74	6.0%
ibm04	17.07	15.79	7.5%
ibm05	42.19	38.57	8.6%
avg			6.88%

Table 8: Effect of the final GP stage on the final placement results.

4.4 Detailed Placement Heuristics

The simulated annealing approach is the most popular DP algorithm used in other placement tools. However, due to the huge computational complexity in the DP phase, simulated annealing at this stage is very slow. For instance, the DP phase of iTools (formerly TimberWolf) which uses simulated annealing consumes more than 80% of the total runtime on large circuits. In Dragon, since there is relatively little work left after GP is done. Instead of widely used simulated annealing, a greedy algorithm is used in the DP phase. Our DP consists of two steps. First, all the overlapping cells are spread out to produce a legal placement. Then cells are exchanged locally to reduce wirelength. In the cell spreading step, all the cells belonging to the same global bin will be re-arranged from left to right using a random order. After the cell spreading step, the greedy cell exchange algorithm is used to further reduce wirelength. The algorithm randomly chooses a base cell. Then it decides whether to perform a vertical search or a horizontal search by using a adjustable parameter R_v. R_v is the ratio of vertical searches and satisfy $0 \leq R_v \leq 1$. If a vertical search is picked to perform, the cell directly above or below the base cell will be picked as the target cell. The base cell and the target cell will be exchanged to see if this exchange result in a reduction in wirelength. If the size of the base cell is not the same as the size of the target cell, horizontal positions of all the cells to the right of the base and the target cell need to be adjusted to remove the overlap and whitespace. If a horizontal search is picked to perform, all the $W - 1$ cells to the left or right of the base cell will be picked as target cells. All the target cells and the base cell itself forms a window containing W cells. W is

another adjustable parameter called "window size". All the target cells and the base cells will be re-arranged horizontally in an exhaustive search manner to look for possible reduction in wirelength. In the horizontal search, no matter how we re-arrange the cell order inside the window, the total width of the window remains unchanged. No cells outside the window need to be moved during the horizontal search. Thus the horizontal search is computationally much cheaper than the vertical search. R_v being 50% can balance the search performed in both directions but has a higher computational cost than a smaller R_v values. Similarly, having a large window size W will explore more solution spaces but increase the runtime. We empirically find the right value for R_v and W. Figure 5 shows the relationship between values of R_v and W and the final placement quality. It shows that placement quality does not improve much when $W \geq 5$ and $R_v \geq 20\%$. Therefore, we set $W = 5$ and $R_v = 20\%$ in our DP.

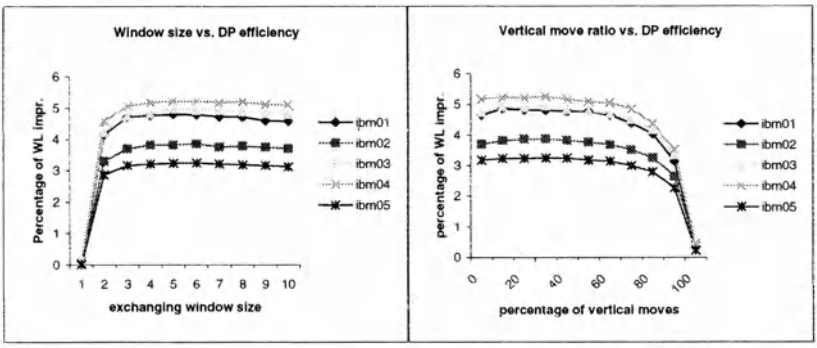


Figure 5: Effects of detailed placement parameters.

5 Experimental Results

Dragon was implemented in C++. All the experiments were performed on a 500 MHz PC running under the Solaris-x86 operating system. We picked five large circuits from MCNC suite and eight large circuits from ISPD98 suite as our testing circuits. MCNC circuits are picked because they have been widely used in literature. However, all MCNC circuits except golem3 are too small (< 30,000 cells). The eight ISPD98 circuits we picked are relatively large (from 60,000 to 200,000 cells). However, they are not suitable for standard cell placement due to the existence of very large cells in these

circuits. The width of the largest cell in these circuits could be longer than the width of the layout area. We modify these ISPD98 circuits by removing large cells. Specifically, we remove cells with the area larger than twenty times the area of the smallest cell in the circuit. The degree of a net might decrease due to removing of large cells. Table 5 shows the characteristics of the testing circuits. We compare Dragon with iTools1.4.0 [5] on all testing circuits. Since we do not have the access to other older placement tools, we use numbers reported in literature on MCNC circuits for comparison. Since ISPD98 circuits have not been used for placement before, only Dragon and iTools1.4.0 are compared on ISPD98 circuits.

Ckts	#cells	#nets	#pins	#rows
industry2	12142	13419	125555	72
industry3	15059	21940	176584	54
avq.small	21854	22124	82601	80
avq.large	25114	25384	82751	86
golem3	99932	143379	336299	128
ibm11	68119	78843	248889	128
ibm12	69026	75157	301604	128
ibm13	81018	97574	311403	128
ibm14	147088	147605	547333	128
ibm15	157861	183684	653684	128
ibm16	181633	188324	762218	128
ibm17	182359	186764	834953	128
ibm18	210323	201560	817331	128

Table 9: Properties of MCNC and ISPD98 benchmark circuits.

Table 10 shows the placement results of Dragon, iTools1.4.0, TimberWolf7.0 and TUM [27] for MCNC circuits. Wirelength results of TimberWolf7.0 and TUM are obtained from [28] and [27], respectively. Runtime comparison is very difficult since different machines were used in literature. Therefore we do not report the runtime for TimberWolf7.0 and TUM. On small MCNC circuits (less than 30,000 cells), Dragon uses less time than iTools but the wirelength results are about 5% worse. However, it still outperforms other successful university tools like TimberWolf7.0 and TUM.

Table 11 shows the placement results of Dragon and iTools1.4.0 on large testing circuits which has more than 60,000 cells including eight ISPD98

[5] formerly TimberWolf, http://www.internetcad.com

Ckts	TW7.0	TUM	iTools1.4.0		Dragon	
	WL	WL	WL	time	WL	time
industry2	13.53	14.6	12.30	1537	12.88	1461
industry3	42.84	45.1	40.13	3154	42.33	2849
avq.small	5.41	4.91	4.84	1915	5.17	1420
avq.large	5.86	5.38	5.19	2043	5.25	1984
golem3	90.39	-	85.44	24380	77.56	8422

Table 10: MCNC circuits comparison.

circuits and one MCNC circuit. On average, Dragon produces placement results with the same quality as iTools for these circuits while spending much less time (1.9× speedup). We also observed that Dragon performs better on circuits larger than 100,000 cells (1.4% better results and 2.1× speedup). This is the first time that a university placement tool can produce good results on large industrial circuits.

Ckts	iTools1.4.0		Dragon		Comparison	
	WL	time	WL	time	impr.	spdup
ibm11	39.76	18251	40.82	10301	-2.6%	1.8×
ibm12	69.56	18075	70.38	14198	-1.2%	1.3×
ibm13	49.11	22577	51.02	15456	-3.9%	1.5×
ibm14	118.8	43057	118.0	31894	0.7%	1.4×
ibm15	130.6	54262	130.8	22808	0.0%	2.4×
ibm16	163.8	70320	168.8	39001	-3.0%	1.8×
ibm17	256.6	72094	255.1	38752	0.5%	1.9×
ibm18	191.7	75363	189.6	39603	1.1%	1.9×
golem3	85.44	24380	77.56	8422	9.2%	2.9×
ave					0.1%	1.9×
ave*					1.4%	2.1×

Table 11: Placement results for circuits larger than 60,000 cells (* average value for circuits larger than 100k cells).

6 Rent Exponent Analysis and Wirelength Estimation

One of the important topics in VLSI layout generation is to estimate wirelength before physical design. An accurate estimate of total wirelength or wirelength distribution would greatly benefit logic designers, since some layout features (routability, die size) could be obtained much earlier once the wirelength is known. Over the past two decades, lots of wirelength estimation approaches have been proposed. Some of them adopt neighborhood analysis, more approaches are originated from the well known Rent's rule.

Rent's rule was first described by Landman and Russo in 1971 [21]. It relates the number of external connections and the number of cells for a given block in a partitioned circuit. Rent's rule has been observed on many of real designs. It has extensive applications in VLSI design. A priori wirelength estimation is one of the most important applications of Rent's rule. The classical work [29, 30] gives good estimates for post layout interconnect wirelength. More recent work improves the estimation by considering *occupying probability* [31] or recursively applying Rent's rule throughout an entire monolithic system [32]. Extension of basic wirelength estimation, including routing utilization estimation [33], congestion estimation [34], 3-D design performance analysis [35, 36], interconnect fan-out distribution [37], are also of value for physical design automation tools.

The Rent's rule correlation is commonly presented by $P = kG^r$, where P and G are the number of external nets and the number of cells for a block, respectively. k is often called Rent coefficient, which is the average number of pins per cell. The Rent exponent, r, is the feature parameter of the circuit. Hagen, et. al., studied Rent Exponent of circuits based on comparison of different partitioning approaches [38], proposed the *intrinsic Rent exponent* which indicates the minimum Rent exponent obtained by an optimal partitioning method. Furthermore, it is argued that the Rent exponent is a measure of partitioning approaches. Other related work includes the proposal of Region III [39], the local variation of Rent exponent [40] and Rent exponent prediction [41].

One of the fundamental issues in Rent's rule study is the extraction of Rent exponent for a given circuit. Traditionally, Rent exponents were obtained by partitioning circuits and analyzing the partitioned subcircuits. In [21] a multi-way partitioning algorithm is used to generate partitioning instances. For each instance, the average subcircuit size and the average number of pins per subcircuit are calculated and the result represents a

data point on a log-log scale. A linear regression is then applied to find the slope of the fitted line, which is the Rent exponent of the circuit. Similar strategy was employed in [38] in which a recursive bipartitioning approach is used to generate partitioning instances.

In this section we propose a different method of extracting Rent exponent for a given circuit, that is, achieving Rent exponent from an existing placement. This is to better understand the notion of Rent parameters, and is not to suggest the Rent parameters should be obtained from placement. We argue that Rent exponents extracted from partitioning and placement are not identical. However, there exists a relationship between these two exponents. We experimentally evaluate this relationship. Experiments have been done on mid-size or large benchmark circuits in order to provide useful information. close to real world. To take into account the variety of placement tools, three recent university placement tools (*Capo* [17], *Feng Shui* [18] and *Dragon* [19]) are used in this work. There is no doubt that extracting Rent exponent from placement is much slower than from partitioning. Furthermore, Rent exponent is indeed useless after placement stage. However, studies on this issue will provide a different point of view on Rent's rule and its applications.

6.1 Rent Exponents for Partitioning and Placement

Conventional methods of extracting Rent exponent are based on partitioning. Analyzing an existing placement of a circuit, however, will give a new way of measuring Rent exponent. It is not surprise that the Rent exponents obtained from two methods are different. Partitioning based extraction focuses on the topological structure of the circuit, while placement based extraction concentrates on the geometrical information of the placed circuit. Figure 6, algorithm 1 and algorithm 2 explain the two different methods to extract Rent exponent.

In the first method extract-Rent-by-Partitioning, a partitioning algorithm is used to recursively bipartition the original circuits. At each bipartitioning level, the average number of cells and average number of external nets for all subcircuits are recorded. The pair of numbers form a point on a log-log plane. After achieving enough points, a linear regression is performed to obtain Rent exponent.

To extract Rent exponent from placement, we first place the circuit using existing placement tools. Then we divide the layout area into several regions, analyze the subcircuit in each region. The average number of cells and average number of external nets for all regions are recorded. This dividing

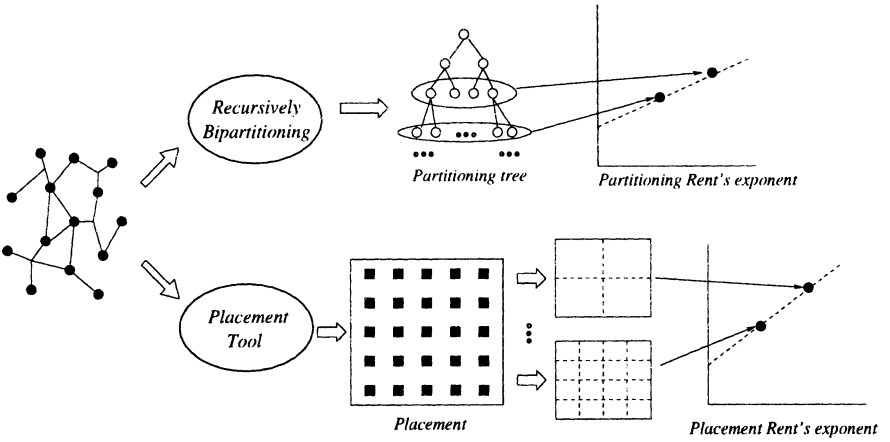


Figure 6: Rent exponent extraction from recursive bipartitioning (upper half) and placement (lower half)

Algorithm 1 Extract-Rent-by-Partitioning(G)

Input: Circuit $G = (V, E)$
Output: Rent exponent r

1. Recursively bipartition the original circuits. At each recursive level, calculate the average number of cells per partition and the average number of external nets over all partitions. Save the data pair to (B_i, P_i). i is the depth of recursive partitioning. Partitioning stops when reaching a given depth n.

2. Apply linear regression on the log-log scaled data pairs: (B_k, P_k), (B_{k+1}, P_{k+1}), ..., (B_n, P_n) (k is a given number around 4-6 to skip Region II)

3. Return the slope of the fitted line by linear regression.

Algorithm 2 Extract-Rent-by-Placement(G)

Input: Circuit $G = (V, E)$
Output: Rent exponent r'

Place the circuit on two dimensional plane,
for $i \leftarrow 1$ to a given depth n **do**
  Divide the core cell area into 2^i regular regions,
  cell group of a region consists of cells which are placed into this region,
  Compute the average number of cells per region partition and the average number of external nets over all regions.
  Save the data pair to (B_i, P_i).
end for
Apply linear regression on the log-log scaled data pairs: (B_k, P_k), (B_{k+1}, P_{k+1}), ..., (B_n, P_n) (k is a given number around 4-6 to skip Region II)
Return the slope of the fitted line by linear regression

step continues to a given depth. Then we obtain Rent exponent by linear regression on the recorded points.

A detailed step of implementing *Extract-Rent-by-Partitioning* is as follows: when a subcircuit is partitioned into two smaller subcircuits, the nets which connect the outside cells are not considered. For multi-terminal nets, part of the net will be reserved and the external pins are ignored.

Definition 6.1 *For a given circuit and a bipartition approach, the partitioning Rent exponent r is the output of the algorithm Extract-Rent-by-Partitioning()*

Definition 6.2 *For a given circuit and a wirelength optimized placement of the circuit, the placement Rent exponent r' is the output of the algorithm Extract-Rent-by-Placement()*

Since partitioning and placement are related problems, the partitioning Rent exponent and placement Rent exponent might also be related. Partitioning tends to minimize the number of cut nets for two subcircuits, which in turn leads to a small number of external nets for a subcircuit. While in a wirelength driven placement, the cells which are tightly connected are placed closer. There is no effort on reducing the crossing nets between two regions. As shown in Figure 7, for a given size (B_1) of the subcircuit, the number of external nets for placement is larger than that for partitioning.

Both straight lines share the same intersection point. Therefore the slope of the line which is obtained by partitioning is smaller than the slope of the other line, which is done by placement.

$$r < r'$$

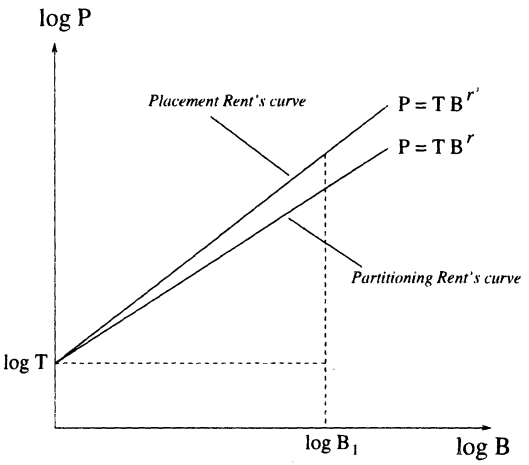


Figure 7: Comparison between partitioning Rent exponent and placement Rent exponent

If the placement engine is a min-cut class approach, we can derive a relationship between the two Rent exponents. Figure 8 illustrates two different bipartitioning problems. In Figure 8(a), the partitioner only considers the interconnects between cells of the subcircuit to be partitioned. We call this problem the *pure* bipartitioning problem. In Figure 8(b), external nets, which connect cells of this subcircuit to other subcircuits, are also included into partitioning problem. This is the bipartitioning problem with *terminal propagation*, which is normally used in min-cut class placement tools, as shown in Figure 8(c). It is the difference between these two bipartitioning approaches which explains the difference between partitioning Rent exponent and placement Rent exponent.

In the pure bipartitioning problem without terminal propagation, assuming the sizes of the subcircuits are B_1 and B_2. The minimized number of net cuts is denoted by C (Figure 8(a)). For the bipartitioning process with terminal propagation, let C' be the number of cut nets of bipartitioning. We have $C' > C$ because of the effect of the external nets. According to Rent's rule, from Figure 8(a), we obtain:

$$P_1 + C = P = TB_1^r, \tag{9}$$

where P is the total number of external nets for subcircuit B_1. P_1 is the number of the external nets which are not cut nets. T is Rent coefficient, the average number of pins per cell. r is the partitioning Rent exponent.

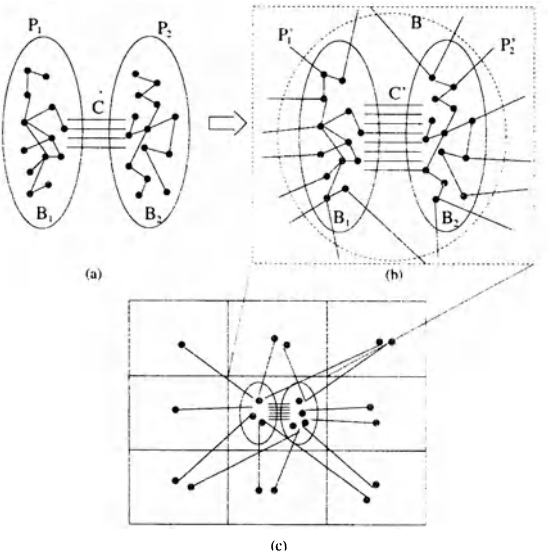


Figure 8: Comparison between a pure partitioning (a) and a partitioning with terminal propagation in min-cut placement (b), (c). The former only consider the internal nets, while the latter consider both internal nets and external nets.

We assume that all the nets are two-terminal nets. Applying Rent's rule on the original subcircuit before partitioning, we obtain:

$$P_1 + P_2 = T(B_1 + B_2)^r \tag{10}$$

For simplicity, we assume that in a balanced bipartitioning, $B_1 = B_2$ and $P_1 = P_2$. From equation (9) and (10), we have:

$$P_1 = 2^{r-1}P$$

In the bipartitioning with terminal propagation (Figure 8(b)), there are P_1 external nets connected to other subcircuits. These nets connect to cells

that are located either to the left or to the right of original circuit. The external nets connected to the right side ($P_1/2$ nets) will "drag" cells from left to right, thus they may increase the cut nets of the partitioning. We assume that one such external net increases the number of cut nets by α. α is a real number between 0 and 1. It represents the possibility that an external net increases the number of cut nets by one.

The same situation exists on the right subcircuit. Thus the result of partitioning with terminal propagation will increase by αP_1. Therefore for a partitioned subcircuit, the number of external nets P' after terminal propagation based partitioning is:

$$P' = P + \alpha P_1 = (1 + \alpha \cdot 2^{r-1})P$$

Since $P = TB_1^r$ and $P' = TB_1^{r'}$ (r and r' are partitioning Rent exponent and placement Rent exponent, respectively), we have,

$$r' - r = \frac{logP' - logT}{logB_1} - \frac{logP - logT}{logB_1}$$

$$= \frac{log(P'/P)}{logB_1} = \frac{log(1 + \alpha \cdot 2^{r-1})}{logB_1}$$

$$r' = r + \frac{log(1 + \alpha \cdot 2^{r-1})}{logB_1} \tag{11}$$

where B_1 is the number of cells of the subcircuit. α is a various parameter with different values in different hierarchical levels. In practice we set B_1 to $|V|/2^5$ and α to 1 to avoid the Rent's rule region II [6].

Equation (11) shows that the placement Rent exponent (r') is larger than the partitioning Rent exponent (r). It should be noted that the analysis is based on some simplifications (e.g. two-terminal nets). The valid range of Equation (11) is limited. For example, if r is close either 0 or 1, the equation does not give meaningful result. However, for ordinary circuits and ordinary partitioning Rent exponents, this equation approximately yields a placement Rent exponent which can be used for estimation purpose.

[6]Region II corresponds a few top-most levels of the partitioning or placement where the number of cells and the number of external nets don't follow the Rent's rule.

6.2 Experimental Validation

Equation (11) shows that we can derive placement Rent exponent r' from the partitioning Rent exponent r. The following experiments are performed to evaluate the relationship.

We experimentally extract both partitioning exponent and placement exponent for a set of circuits. The circuits are chosen from MCNC and IBM-PLACE benchmark suits. IBM-PLACE benchmarks are derived from ISPD98 IBM partitioning benchmark suits [20]. Experimental circuit sizes range from 21,000 cells to 210,000 cells. For partitioning Rent exponent, we use hMetis [15] as the partitioning tool. Unbalance factor is set to 1% in each bipartitioning call. For placement Rent exponent, three different placement tools are used. They are *Capo* [17], *Feng Shui* [18] and *Dragon* [19]. The first two are based on min-cut placement method. Dragon employs both cut and wirelength minimization in hierarchical placement flow. All the experiments are performed on Sun workstations with 400MHz CPU and 128M memory. The depths of both extract-Rent-by-partitioning and extract-Rent-by-placement is set to 14, i.e., 14 data points are collected from partitioning or placement. First 5 points are discarded in order to avoid effects by Rent's rule region II. The linear regression is performed on 9 data points for each benchmark.

Figure 9 shows a sample extraction on ibm15 benchmark. The lower line is the result of linear regression on data points by recursive bipartitioning. Three upper lines are extracted from placement outputs by *Capo*, *Feng Shui* and *Dragon*. All the slopes of three upper lines are larger than the slope of the "partitioning line", supporting the relationship between partitioning Rent exponent and placement Rent exponent mentioned in section 2.

Table 12 shows the comparison between partitioning Rent exponent r, estimated placement Rent exponent r' which is derived from equation (11), and three real placement Rent exponents r'' by different placement tools. Figure 10 gives a better view of comparison. Note that Rent exponents of different placement techniques are different. However, they do not vary much for a given circuit.

6.3 Wirelength Estimation Using Rent Exponents

In Section 6.1 we discussed the difference between the partitioning Rent exponent and the placement Rent exponent. In wirelength estimation, the total wirelength or the average wirelength is a function of Rent exponent. Different Rent exponents result in different estimates. In order to obtain

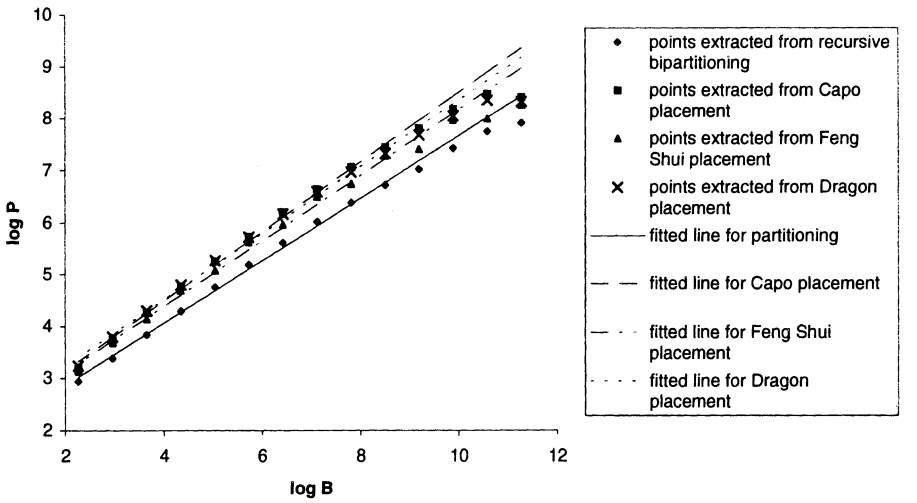


Figure 9: Rent's rule fitted line based on partitioning and placement for circuit ibm15. The lower line is the result of linear regression on data points by recursive bipartitioning. Three upper lines are from placement outputs.

ckt	*#cells*	*#nets*	Partition Rent r	Estimated Place r'	Placement Rent r''		
					Capo	*Feng Shui*	*Dragon*
avq.small	21,854	22,124	0.449	0.532	0.599	0.533	0.548
avq.large	25,114	25,384	0.449	0.530	0.596	0.538	0.543
golem3	99,932	143,379	0.556	0.617	0.615	0.600	0.575
ibm11	68,119	78,843	0.608	0.680	0.693	0.682	0.667
ibm12	69,026	75,157	0.648	0.723	0.708	0.694	0.683
ibm13	81,018	97,574	0.600	0.674	0.689	0.668	0.648
ibm14	147,088	147,605	0.622	0.690	0.679	0.650	0.663
ibm15	157,861	183,684	0.599	0.672	0.669	0.630	0.640
ibm16	181,633	188,324	0.609	0.673	0.674	0.627	0.651
ibm17	182,359	186,764	0.645	0.708	0.704	0.650	0.671
ibm18	210,323	201,560	0.600	0.664	0.658	0.615	0.632

Table 12: Comparison between partitioning Rent exponent r, estimated placement Rent exponent r' and real placement Rent exponent r'' extracted from three placement tools' outputs

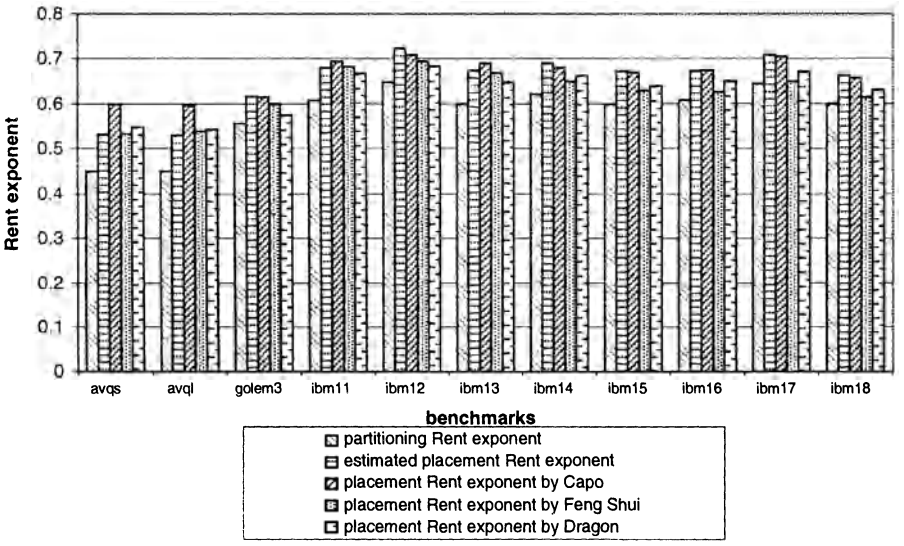


Figure 10: Comparison between partitioning Rent exponent, estimated placement Rent exponent and three real placement Rent exponents by three placement tools

more accurate wirelength estimates, a *proper* Rent exponent is demanded in wirelength estimation approach.

Different Rent Exponents in Estimation

The authors in [38] show that the Rent exponent of a circuit depends on the approach used to derive it. Different Rent exponents are obtained when using different partitioning heuristics. The partitioning Rent exponent, as described in Section 6.1, is derived using a given partitioning approach (hMetis). If we use other partitioning heuristics, or even use different calling parameters[7], different Rent exponents will be obtained. In general these values are relatively close.

Similar situation exists in extracting placement Rent exponent. If we use different placement algorithms and extract Rent exponents from the placement results, we will obtain different placement Rent exponents. Likewise, it is expected that the placement Rent exponents do not have much variation from different placement algorithms.

[7]Unbalance factor, number of starts, option to minimize number of external pins.

In the wirelength estimation work [29, 31], the authors adopt a hierarchical placement model and assume that Rent's rule holds for all subcircuits at each hierarchical level. In [32] the wirelength distribution is derived from the number of interconnects between gates that are a given distance away, which is determined using Rent's rule. Note that in above approaches, partitioning Rent exponent and placement Rent exponent are not distinguished from each other. By definition, wirelength estimation requires the placement Rent exponent, while in the real world the partitioning Rent exponent may be used to do the calculation. This is because the partitioning Rent exponent is easy to obtain simply by recursively bipartitioning the circuit. For small circuits, this ambiguity between two Rent exponents does not have much influence on wirelength estimates since the two Rent exponents are close. However, for larger circuits, when the partitioning Rent exponent is considerably different from the placement Rent exponent, wirelength estimates using the partitioning Rent exponent tends to under-estimate the total wirelength. This under-estimation is verified by the following experiments.

In Section 6.2 we obtained the partitioning Rent exponent and three placement Rent exponents for each circuit. With these exponents, we estimate the total wirelength based on existing wirelength distribution models. Both classic Donath's method [29] and the recent Davis's wirelength distribution model [32] are used in this approach.

The estimation results are compared with real wirelength given by the global router. Since we have three placement outputs, we also have three corresponding global routing results. For the sake of simplicity, the number of rows in standard cell placement is set to an integer which is a power of 2 (128 in the experiments). We also assume that the grid in global routing is a square with unit width and unit height. For better comparison, the estimated total wirelength is scaled to the total length in terms of global routing grid units. Specifically, if the number of cells in a circuit is G, and the global routing grid is $n \times n$, then the scaled estimated wirelength WL is,

$$WL = WL' \frac{n}{\sqrt{G}}$$

where WL' is the estimated wirelength in grid units.

Table 13 shows a comparison between the estimated wirelength and real wirelength after global routing. For each circuit, two estimation methods (Donath's and Davis's) are used on four Rent exponents (one partitioning Rent exponent and three placement Rent exponents). Placements of cir-

cuit are obtained using three different placement tools. For each placement output the corresponding global routing result is shown.

ckt	Partition			Placement				
	r	Donath's Est.	Davis's Est.	place tool	r''	Donath's Est.	Davis's Est.	Real WL
ibm11	0.608	562	442	*Capo*	0.693	800	583	489
				FS	0.682	764	562	494
				Dragon	0.667	718	534	483
ibm12	0.648	798	572	*Capo*	0.708	1024	699	738
				FS	0.694	967	667	744
				Dragon	0.683	798	572	646
ibm13	0.600	640	500	*Capo*	0.689	937	675	676
				FS	0.668	856	627	716
				Dragon	0.648	786	586	605
ibm14	0.622	972	752	*Capo*	0.679	1273	935	913
				FS	0.650	1109	836	841
				Dragon	0.663	1180	879	822
ibm15	0.599	1062	807	*Capo*	0.669	1482	1053	1242
				FS	0.630	1227	905	1175
				Dragon	0.640	1288	940	1196
ibm16	0.609	1236	925	*Capo*	0.674	1694	1192	1207
				FS	0.627	1349	991	1090
				Dragon	0.651	1512	1086	1078
ibm17	0.645	1606	1123	*Capo*	0.704	2149	1429	1661
				FS	0.650	1646	1146	1651
				Dragon	0.671	1826	1247	1653
ibm18	0.600	1207	933	*Capo*	0.658	1411	1056	1108
				FS	0.615	1605	1173	1247
				Dragon	0.632	1300	989	1090

Table 13: Partitioning Rent exponent r and wirelength estimates by two estimation methods (Donath's and Davis's), comparing with placement exponent r'' by three different placement tools (Capo, FS(Feng Shui) and Dragon), and the wirelength(WL) estimates based on r''. The final column is the real wirelength output by global router. Both estimated and real wirelengths are in 10^3 grid units of global routing.

It is well accepted that Donath's classic work over-estimates the total wirelength for most circuits. Therefore we focus on wirelength estimates by Davis's wirelength distribution model. From Table 13 we observe that the wirelength estimates based on the partitioning Rent exponent are always smaller then the real wirelength. While wirelength estimates based on the placement Rent exponents are more close to the real results. This obser-

vation supports the previous assumption that wirelength estimation should use placement Rent exponent, not the partitioning Rent exponent.

Wirelength Estimation Using Placement Rent Exponent

Table 14 illustrates the wirelength estimation using estimated placement Rent exponent, compared with real wirelength after global routing. The motivation is to estimate wirelength using the partitioning Rent exponent, as it is easier to obtain partitioning Rent exponent. For most circuits, wirelength estimates using estimated placement Rent exponent are close to real wirelength by placement tool and global router. However, it is not clear whether this estimation method is more accurate or not. Wirelength result varies in different placement tools. In addition, global router's parameters (for instance, routing capacity) also affect total routed wirelength. A good wirelength estimate is only meaningful in a given context. In general there is no *perfect* wirelength estimation independent of place and route tool.

			Est.	Real WL ($\times 10^3 units$)		
ckt	r	r'	WL by r'	*Capo*	*Feng Shui*	*Dragon*
ibm11	0.608	0.680	558	489	494	483
ibm12	0.648	0.723	734	738	744	646
ibm13	0.600	0.674	641	676	716	605
ibm14	0.622	0.690	977	913	841	822
ibm15	0.599	0.672	1065	1242	1175	1196
ibm16	0.609	0.673	1189	1207	1090	1078
ibm17	0.645	0.708	1453	1661	1651	1653
ibm18	0.600	0.664	1204	1108	1247	1090

Table 14: Partitioning Rent exponent r, estimated placement Rent exponent r' and estimated total wirelength based on r', comparing with the total global routed wirelength from three placement outputs.

Wirelength and Placement Rent Exponent

In [38] the Rent exponent is regarded as a metric of quality of partitioning algorithm. It is interesting to know whether there is a correlation between the placement quality and the Rent exponent of placement. Previously the quality of placement is measured by the total bounding box wirelength or wirelength after global routing. Therefore we compare placement wirelength and Rent exponents for different placement tools.

Table 15 lists the Rent exponent, total bounding box wirelength and

total routed wirelength for three placement approaches. For consistency, the total bounding box wirelength is reported in grid units of global routing. The global router is based on maze routing including rip-up and re-route. The capacity of global routing edges is set to a value such that the number of nets which are rip-up and re-routed is less than 10% of the total nets. This is to reduce the influence of the global routing on the placement.

Figure 11 shows the comparison more clearly. For most circuits the smaller Rent exponent relates to less total wirelength. Some other circuits show the contrary cases. However, the difference are relatively small in these cases. The correlation exists for both bounding box wirelength and routed wirelength. Thus we conclude that the Rent exponent of placement is a good metric of placement quality.

ckt	Placement Rent exponent			Bounding Box WL ($\times 10^3$ grid units)			Rrouted WL ($\times 10^3$ grid units)		
	Capo	*FS*	*Dragon*	*Capo*	*FS*	*Dragon*	*Capo*	*FS*	*Dragon*
ibm11	0.693	0.682	0.667	435	442	423	489	494	483
ibm12	0.708	0.694	0.683	655	654	567	738	744	646
ibm13	0.689	0.668	0.648	505	510	487	676	716	605
ibm14	0.679	0.650	0.663	827	759	740	913	841	822
ibm15	0.669	0.630	0.640	951	882	890	1242	1175	1196
ibm16	0.674	0.627	0.651	1025	972	961	1207	1090	1078
ibm17	0.704	0.650	0.671	1483	1315	1364	1661	1651	1653
ibm18	0.658	0.615	0.632	1088	953	967	1108	1247	1090

Table 15: Placement Rent exponents derived by three placement tools, with the normalized bounding box wirelength and normalized routed wirelength. Wirelengths are reported in 10^3 units of global routing grid.

7 Conclusion

Top-down hierarchical approach and multi-level hypergraph partitioning technique are two most important components in modern standard-cell placement tools. We use the top-down hierarchical approach to develop a powerful standard cell placement tool, *Dragon*. We theoretically study the properties of using net-cut minimization methods in placement tools. We argue that net-cut minimization is a good and important shortcut to get high quality placement results in the shortest amount of time. With the effective integration of net-cut and wirelength, the GP phase of Dragon is able to produce superior global placement results very fast. Only little amount of

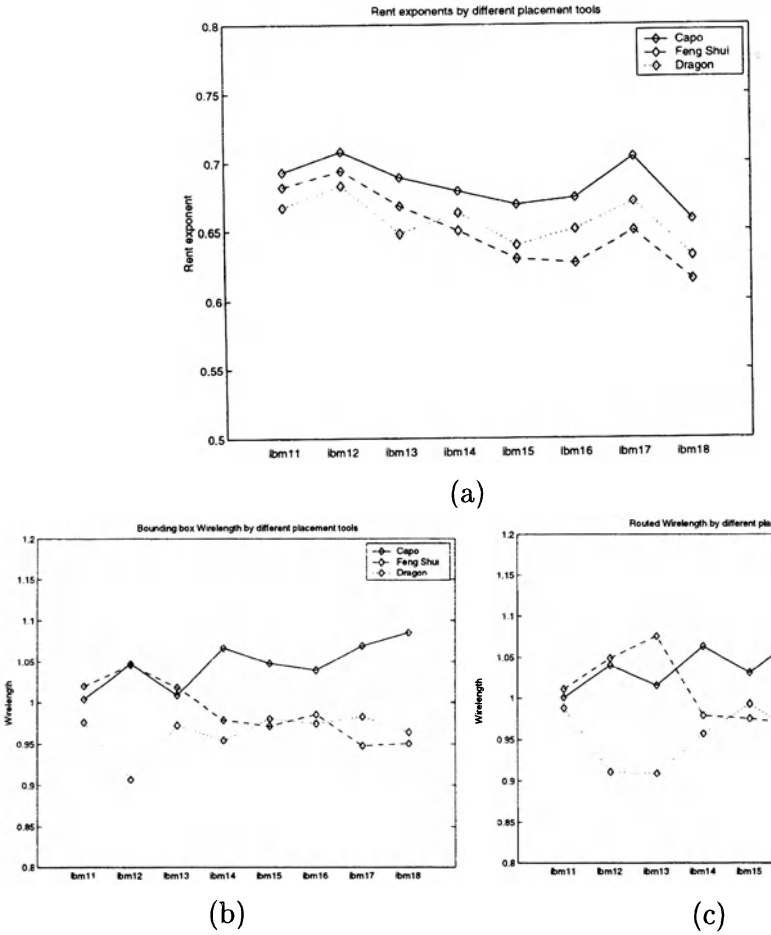


Figure 11: (a) Placement Rent exponents derived from layouts by three different placement tools(*Capo*, *Feng Shui* and *Dragon*). (b) Total bounding box wirelength in grid units by three placement tools. (c) Total routed wirelength in grid units by three placement tools. In (b) and (c) wirelengths are normalized by dividing the average value of three placement tools.

effort is needed in the DP phase. Dragon was tested on recently released large benchmark suite ISPD98 and old MCNC suite. For large circuits which have more than 100,000 cells, Dragon can produce slightly better placement results while spending much less amount of time than the highly optimized commercial iTools. This is the first university tool that produces good results on large publically available circuits.

Wirelength estimation for large circuits is a complex problem. A number of factors can affect the accuracy of estimating, including the approach to obtain Rent exponent, the placement algorithm used in the design flow and the quality or parameters of the global router. In order to obtain accurate wirelength estimates, designers need to adjust estimating model and Rent exponent extraction method according to the place and route tool they employ. Precise wirelength estimation needs extensive experimental data as well as theoretical formulation.

References

[1] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization". *IEEE Transactions on Computer Aided Design*, 10(3):365–365, 1991.

[2] G. Sigl, K. Doll, and F. M. Johannes. "Analytical Placement: A Linear or a Quadratic Objective Function". In *Design Automation Conference*, pages 427–432. IEEE/ACM, 1991.

[3] C. Sechen and K. W. Lee. "An Improved Simulated Annealing Algorithm for Row–Based Placement". In *Design Automation Conference*, pages 180–183. IEEE/ACM, 1988.

[4] D. Huang and A. B. Kahng. "Partitioning-based Standard-cell Global Placement with an Exact Objective". In *International Symposium on Physical Design*, pages 18–25. ACM, April 1997.

[5] M. Sarrafzadeh and M. Wang. "NRG: Global and Detailed Placement". In *International Conference on Computer-Aided Design*. IEEE, November 1997.

[6] W. Donath, R. Norman, B. Agrawal, S. Bello, S. Han, J. Kurtzberg, P. Lowy, and R. McMillan. "Timing Driven Placement Using Complete Path Delays". In *Design Automation Conference*, pages 84–89. IEEE/ACM, 1990.

[7] W. Swartz and C. Sechen. "Timing Driven Placement for Large Standard Cell Circuits". In *Design Automation Conference*, pages 211–215. IEEE/ACM, 1995.

[8] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel. "Chip Layout Optimization Using Critical Path Weighting". In *Design Automation Conference*, pages 133–136. IEEE/ACM, 1984.

[9] M. Burstein and M. N. Youssef. "Timing Influenced Layout Design". In *Design Automation Conference*, pages 124–130, 1985.

[10] R.S. Tsay and J. Koehl. "An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement". In *Design Automation Conference*, pages 620–625. IEEE/ACM, 1991.

[11] P.S. Hauge, R. Nair, and E. J. Yoffa. "Circuit Placement for Predictable Performance". In *International Conference on Computer-Aided Design*, pages 88–91. IEEE, 1987.

[12] G. E. Tellez, D. A. Knol, and M. Sarrafzadeh. "A Performance-Driven Placement Technique Based on a New Budgeting Criterion". In *International Symposium on Circuits and Systems*, pages 504–507. ACM, May 1996.

[13] M. Sarrafzadeh, D. A. Knol, and G. E. Tellez. "Unification of Budgeting and Placement". In *Design Automation Conference*, pages 758–761. IEEE/ACM, 1997.

[14] M. Sarrafzadeh, D. A. Knol, and G. E. Tellez. "A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement". *IEEE Transactions on Computer Aided Design*, 16(11):1332–1341, 1997.

[15] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. "Multilevel Hypergraph Partitioning: Application in VLSI Domain". In *Design Automation Conference*, pages 526–529. IEEE/ACM, 1997.

[16] A. E. Caldwell, A. B. Kahng, A. A. Kennings, and I. L. Markov. "Hypergraph Partitioning for VLSI CAD: Methodology for Reporting, and New Results". In *Design Automation Conference*, pages 349–354. IEEE/ACM, 1999.

[17] A. E. Caldwell, A. B. Kahng, and I. L. Markov. "Can Recursive Bisection Alone Produce Routable Placements?" In *Design Automation Conference*, pages 153–158. IEEE/ACM, June 2000.

[18] M. C. Yildiz and P. H. Madden. "Global Objectives for Standard Cell Placement". to appear in *Proceedings of the Eleventh Great Lakes Symposium on VLSI*, March 2001.

[19] M. Wang, X. Yang, and M. Sarrafzadeh. "Dragon2000: Fast Standard-cell Placement for Large Circuits". In *International Conference on Computer-Aided Design*, pages 260–263. IEEE, 2000.

[20] C. J. Alpert. "The ISPD98 Circuit Benchmark Suite". In *International Symposium on Physical Design*, pages 18–25. ACM, April 1998.

[21] B. Landman and R. Russo. "On a Pin Versus Block Relationship for Partitions of Logic Graphs ". *IEEE Transactions on Computers*, c-20: 1469–1479, 1971.

[22] X. Yang, M. Wang, K. Eguro, and M. Sarrafzadeh. "A Snap-On Placement Tool". In *International Symposium on Physical Design*, pages 153–158. ACM, April 2000.

[23] X. Yang, E. Bozorgzadeh, and M. Sarrafzadeh. "Wirelength Estimation based on Rent Exponents of Partitioning and Placement". to appear in *Proc. International Workshop on System-Level Interconnect Prediction*, April 2001.

[24] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov, and A. Zelikovsky. "On Wirelength Estimations for Row-Based Placement". *IEEE Transactions on Computer Aided Design*, 18(9):445–462, 1999.

[25] G. Karypis and V. Kumar. "Multilevel k-way Hypergraph Partitioning". In *Design Automation Conference*, pages 343–348, 1999.

[26] A. E. Dunlop and B. W. Kernighan. "A Procedure for Placement of Standard Cell VLSI Circuits". *IEEE Transactions on Computer Aided Design*, 4(1):92–98, January 1985.

[27] H. Eisenmann and F. M. Johannes. "Generic Global Placement and Floorplanning". In *Design Automation Conference*, pages 269–274. IEEE/ACM, 1998.

[28] W. J. Sun and C. Sechen. "A Loosely Coupled Parallel Algorithm for Standard Cell Placement ". In *International Conference on Computer-Aided Design*, pages 137–144. IEEE, 1994.

[29] W. E. Donath. "Placement and Average Interconnection Lengths of Computer Logic". *IEEE Transactions on Circuits and Systems*, 26(4):272–277, April 1979.

[30] M. Feuer. "Connectivity of random logic". *IEEE Transactions on Computers*, C-31(1):29–33, Jan 1982.

[31] D. Stroobandt, H. V. Marck, and V. Campenhout. "An Accurate Interconnection Length Estimation for Computer Logic". In *Proceedings of the Sixth Great Lakes Symposium on VLSI*, pages 50–55. IEEE, March 1996.

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

[33] P. Chong and R. K. Brayton. "Estimating and Optimizing Routing Utilization in DSM Design". In *International Workshop on System-Level Interconnect Prediction*. ACM, April 1999.

[34] X. Yang, R. Kastner, and M. Sarrafzadeh. "Congestion Estimation During Top-down Placement". to appear in *Proc. International Symposium on Physical Design*, April 2001.

[35] K. C. Saraswat, S. J. Souri, K. Banerjee, and P. Kapur. "Performance Analysis and Technology of 3-D ICs". In *International Workshop on System-Level Interconnect Prediction*, pages 85–90. ACM, April 2000.

[36] R. Zhang, K. Roy, C. K. Koh, and D. B. Janes. "Stochastic Wire-Length and Delay Distributions of 3-Dimensional Circuits". In *International Conference on Computer-Aided Design*, pages 208–213. IEEE, November 2000.

[37] P. Zarkesh-Ha, J. A. Davis, W. Loh, and J. D. Meindl. "Prediction of Interconnect Fan-Out Distribution Using Rent's Rule". In *International Workshop on System-Level Interconnect Prediction*, pages 107–112. ACM, April 2000.

[38] L. Hagen, A. B. Kahng, F. J. Kurdahi, and C. Ramachandran. "On the Intrinsic Rent Parameter and Spectra-Based Partitioning Methodologies". *IEEE Transactions on Computer Aided Design*, 13(no.1):27–37, Jan 1994.

[39] D. Stroobandt. "On an Efficient Method for Estimating the Interconnection Complexity of Designs and on the Existence of Region III in Rent's Rule". In *Proceedings of the Ninth Great Lakes Symposium on VLSI*, pages 330–331. IEEE, March 1999.

[40] H. V. Marck, D. Stroobandt, and V. Campenhout. "Towards An Extension of Rent's Rule for Describing Local Variations in Interconnection Complexity". In *Proceedings of the Fourth International Conference for Young Computer Scientists*, pages 136–141, 1995.

[41] P. Christie. "Rent Exponent Prediction Methods". *IEEE Transactions on VLSI Systems*, 8(6):679–688, 2000.

Non-Hanan Optimization for Global VLSI Interconnect

Jiang Hu and Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota, Minneapolis, MN 55455
E-mail: {jhu, sachin}@mail.ece.umn.edu

Contents

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 89–123.
© 2001 *Kluwer Academic Publishers.*

1 Introduction

Under current Very Large Scale Integration (VLSI) technology, tens of millions of transistors can be integrated on a single chip, and future trends show that circuit sizes will continue to increase exponentially. Under this scenario, the performance bottleneck will shift to the delays associated with the metal wires, or interconnect, used to join these transistors. In the future, as transistor feature sizes become progressively smaller, the switching speed of a transistor driving a minimum load will become faster. On the other hand, as interconnect wires become thinner and longer, the interconnect delay for global wires is projected to increase, relative to the gate delay, due to the increased wire resistance. Both trends lead interconnect delay to dominate logic delay and become a significant bottleneck in VLSI system performance [1]. As a result, many efforts have been carried out in recent years to improve the interconnect performance, and a good overview of these works is provided in [2–4].

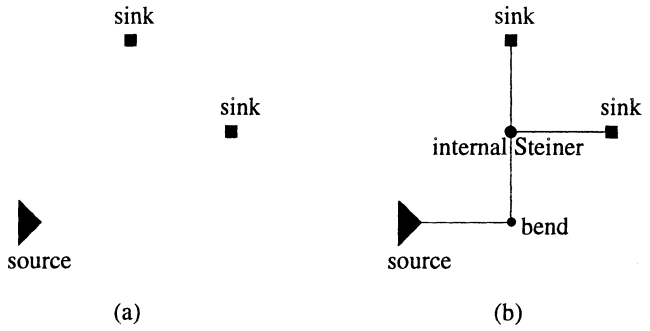


Figure 1: An example of rectilinear Steiner tree for a two sink net.

This chapter addresses the problem of optimizing a single-net global routing tree under stringent timing constraints. A routing net is composed of a source pin from which the signal is sent out, and a set of sink pins where the signal is to be delivered. The single net routing problem is to construct a rectilinear tree representing the wires to span all the pins for a given net. For example, a net with two sinks is shown in Figure 1(a). An internal Steiner node and a bend node are introduced together with four edges in Figure 1(b) to form a rectilinear Steiner tree as a routing solution. The objective in the tree construction is typically to minimize the total wire length and ensure that the delay from source to each sink satisfies the required constraints. The quality of a routing tree can be improved by using a better routing tree topology, appropriately sizing the driver, and by inserting buffers on long

wires to regenerate the signal and to shield non-critical loads from the paths to the critical sinks. All of these techniques will be employed in the work described here.

Considering this routing problem as an optimization problem, the objective is formulated in terms of the total wire length (area) and the routing delay. Minimizing the total wire length is a desirable goal since it can reduce the cost, power consumption and improve the routability, and has been used as the sole optimization objective in the past, when delay was not a significant consideration. For the problem of pure delay reduction, there are many forms in which the objective may be stated, including minimizing either a weighted sum of sink delays, or the maximum delay, or the delay(s) at the critical sink(s). More appropriate formulations that consider both aspects include [5–7], which focus on satisfying the timing specification in an effort to trade off the unnecessary delay reduction into area minimization. In this chapter, we adopt a general formulation of minimizing cost subject to timing constraints, where the cost can either be the wire length or a linear combination of wire length and the buffer/driver cost.

Most traditional works [2–4] restrict the topology space only to points on the Hanan grid [8] (to be defined later), since it simplifies the nature of the problem. Under special problem formulations, where the unconstrained objective is to minimize the wire length or a weighted sum of sink delays [2], it can be proven that an optimal solution can be found by considering only the Hanan grid points as candidate Steiner nodes. However, for the problem of minimizing wire length subject to timing constraints, where both the area and the delay are simultaneously considered during optimization, we will show in Section 3 that the use of Steiner nodes off the Hanan grid is necessary. It is important to note that the tree construction problem that employs these non-Hanan nodes is inherently more difficult than one that considers only Hanan nodes, since the topology search space is much larger. As a prolog, we will first present, in Section 3, an algorithm where the optimal net topology in the non-Hanan space is sought under a constant driver size. A set of delay properties that can aid searching the optimal non-Hanan Steiner nodes efficiently will be developed. Next, in Section 4, we will use this theory as a baseline and expand it to overcome several limiting assumptions. Firstly, we will broaden the set of available optimizations to non-Hanan topology optimization, buffer insertion and driver sizing. In particular, in case of buffer insertion, we consider a realistic situation that buffers are allowed only at limited available spaces. Secondly, we will discuss the importance of employing a higher order delay model for timing critical nets, and will alter our delay metric to use asymptotic waveform evaluation

(AWE) instead of the Elmore delay. Therefore, the algorithms presented in this section constitute practical techniques for minimum cost Steiner tree construction to meet the specified timing constraints. The interconnect optimization methods presented in this chapter are based mostly on the works in [7, 9].

2 Preliminaries

2.1 Basic Definitions

A global net N is specified in terms of a source v_0 and a set of sinks $V_{sink} = \{v_1, v_2, ...v_p\}$. A routing problem for a net N is to find a set of Steiner nodes $V_{Steiner} = \{v_{p+1}, v_{p+2}, ...v_{p+q}\}$ and a set of edges $E = \{e_1, e_2, ...e_{p+q}\}$ to construct a rectilinear Steiner tree (RST) $T(V, E)$, where $V = \{v_0\} \cup V_{sink} \cup V_{Steiner}$, such that E spans all of the nodes in V. The traditional definition of $V_{Steiner}$ includes two types of nodes: (i) *internal Steiner nodes* of degree three or four, denoted by the set $V_{internal}$, and (ii) *bend nodes* of degree two that denote a path switch between a horizontal and a vertical direction, denoted by the set V_{bend}. For example, a net with two sinks is given in Figure 1(a). An internal Steiner node and a bend node are introduced together with four edges in Figure 1(b) to form a rectilinear Steiner tree as a routing solution. A bend node is introduced to make the tree conform to the requirement of rectilinear space, and an internal Steiner node is usually employed to reduce wire length. The location for a node v_j is specified by its coordinates (x_j, y_j), and an edge in E is uniquely identified by the node pair (v_j, v_k), and is denoted as e_{jk} or e_k interchangeably. Note that we assume that v_k is the downstream end of e_{jk}. The edge length l_{jk} is given by the Manhattan distance between the two nodes, which is $|x_j - x_k| + |y_j - y_k|$.

A basic concept that is used in this work is the notion of delay violation. If the delay at an arbitrary sink v_a is $t(v_a)$ and the its required arrival time is $q(v_a)$, then the delay violation $\Delta(v_a) = t(v_a) - q(v_a)$. In order to make a routing tree to satisfy timing constraints, the delay violation of every sink in the net should be non-positive.

2.2 Delay Models

We use the π RC model for a wire segment of length l, as shown in Figure 2, where r and c are resistance and capacitance per unit length, respectively. One popular delay model for interconnect is Elmore model [10]. We express driver resistance as R_d and let C_i denote the total downstream capacitance

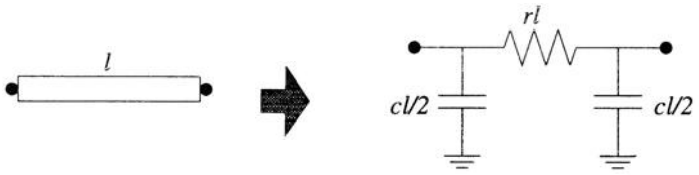


Figure 2: Model of a wire segment.

seen from node v_i. The Elmore delay from driver to a sink v_k is given as:

$$t_k = R_d C_0 + \sum_{e_{ij} \in path(v_0, v_k)} rl_{ij}(\frac{cl_{ij}}{2} + C_j) \quad (1)$$

The Elmore delay model has been widely used in many research works due to its simplicity and high fidelity [11]. Its simplicity not only removes the need for large amount of computation, but also provides a platform on which many theoretical properties can be derived and exploited.

For the routing of critical nets whose timing constraint is stringent, we employ a fourth order AWE model [12]. The fourth order AWE model takes higher order moment information into consideration and can provide a much better accuracy, though the computation time becomes longer. The reason for choosing the fourth order will be explained in more details in Section 4.1.

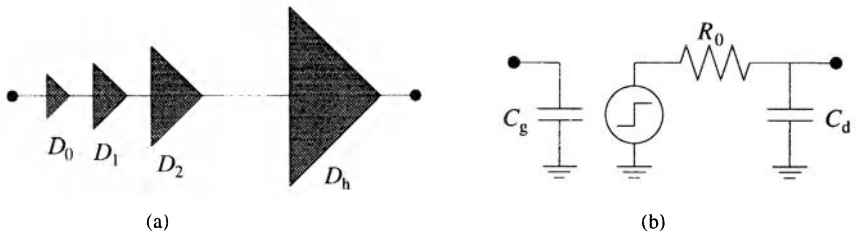


Figure 3: Cascaded drivers and driver model.

For driver sizing problem, we consider the situation where the signal net is driven by a series of cascaded drivers $D_0, D_1, D_2....D_h$ as in Figure 3(a). The driver D_0 is minimum sized and will not be changed in driver sizing. The driver and buffer model that we will use is shown in Figure 3(b). We denote the gate and drain capacitance of D_0 as C_g and C_d, respectively. The interconnect delay among these drivers is typically small and is neglected. The driver resistance and capacitance are assumed to change linearly with respect to the size of driver.

2.3 Soft Edges

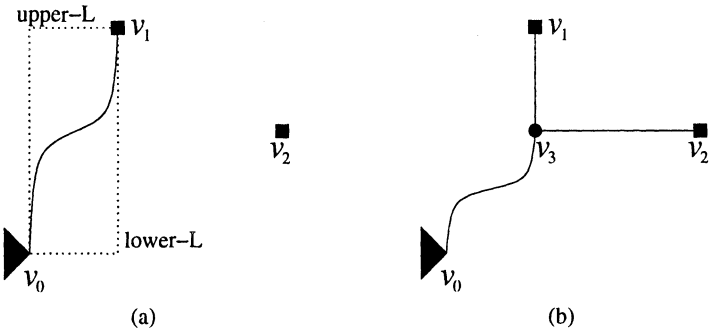


Figure 4: Routing with soft edges.

In the process of routing for one net, multiple options are sometimes available and it is not obvious which of these is the best. Consider the example in Figure 4, where a source v_0 and two sinks v_1 and v_2 are given, and a minimum Steiner tree is to be constructed on this node set by adding one node to the tree at a time. Since a routing tree is built in rectilinear space, each edge must be either horizontal or vertical. If we begin by connecting v_1 to v_0, there are two L-shaped connection options, shown by the dotted lines in Figure 4(a); one bend is required for each connection. The delay and wire length from v_0 to v_1 are same in these two options and it is hard to see which is better at this stage. Instead of fixing the edge orientation immediately as in usual approaches, we defer this decision-making to a stage when the effects of the these options can be discriminated. Here, we formalize this by introducing another type of edge, a *soft edge*, whose route is not specified until there is an obvious better choice.

Definition 2.1 (soft edge) *A soft edge is an edge connecting two nodes* $v_i, v_j \in V$, *such that:*

1. $x_i \neq x_j$ *and* $y_i \neq y_j$,
2. *its edge length* $l_{ij} = |x_i - x_j| + |y_i - y_j|$,
3. *the precise edge route between* v_i *and* v_j *is not determined.*

We will refer to the traditional edges in a rectilinear tree with fixed orientations as *solid edges.* The soft edge connection between v_0 and v_1 is shown by the solid curve in Figure 4(a). In order to minimize wire length, the sink v_2 is connected to the routing tree at the closest connection (CC) point, defined below, between v_2 and edge e_{01}.

Definition 2.2 (*CC* point) *The closest connection (CC) point between a node v_k and an edge e_{ij} is defined by its coordinates x_{CC} and y_{CC} such that:*

$$x_{CC} = median(x_i, x_j, x_k) \text{ and } y_{CC} = median(y_i, y_j, y_k).$$

Note that in Definition 2.2, the edge e_{ij} can be either a soft edge or a solid edge. If the *CC* point does not coincide with either of v_i, v_j and v_k, a Steiner node is introduced at the *CC* point. In the example of Figure 4, Steiner node v_3 is introduced. After this connection has been made, edge e_{31} and e_{32} are solid edges.

At this stage, it can be seen that lower-L is a better choice for connecting v_0 and v_1 than upper-L, since it provides a shorter wire length which is a result reached through deferred decision making. The advantage of using soft edges is that they provide a set of flexible connection choices for subsequent routing steps and avoid premature suboptimal decisions. In fact, we do not need to choose the lower-L when connecting v_0 and v_1, and we may keep the edge e_{03} soft when v_2 is joined as indicated in Figure 4(b). We will show later that the use of soft edges also has advantages that aid utilizing buffer spaces.

3 Non-Hanan Routing

3.1 The Motivation for Using Non-Hanan Points in Global Routing

Drawing horizontal and vertical lines through all the pins in a given net results in the *Hanan grid* [8], illustrated in Figure 5(a). It is proved that there is always an RST with minimum wire length embedded in the Hanan grid as shown in 5(b) [8]. In the situation where the interconnect is purely capacitive and has negligible resistance, the minimum delay tree is identical to the minimum length Steiner tree, since minimizing the length is equivalent to minimizing the delay. When the resistance of the interconnect is appreciable, the type of route is determined by the stringency of the timing specifications.

An appropriate problem formulation for the timing-driven routing problem is to minimize the total wire length subject to timing constraints. If the timing specifications are very loose, then the solution to this problem would be to minimize the total tree length, and would result in a minimum-length Steiner tree. If the timing specifications are extremely tight, and the resistance of the driver is much less than the resistance of the interconnect, a

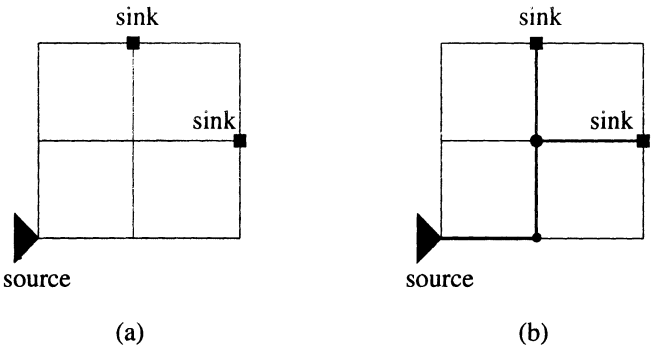


Figure 5: An example of Hanan grid and a minimum RST over the Hanan grid.

star-like topology is generated. The star-like topology emanates at the root of the tree and has direct connections from the root to each sink; this corresponds to the maximum length for the tree. For intermediate specifications, an intermediate solution between the minimum length Steiner tree and the star-like topology is optimal. This is illustrated in Figure 6.

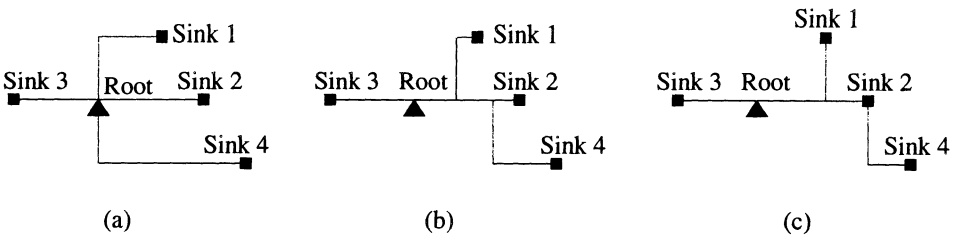


Figure 6: Three scenarios for building a Steiner tree (a) Star-like topology (b) Intermediate topology (c) Minimum-length topology.

In [11], it was proved that only points on the Hanan grid need be considered while solving the problem of minimizing the weighted sum of critical sink delays. For the minmax problem of minimizing the maximum sink delay, it was shown in [11] that it is possible to build a better solution by considering points off the Hanan grid, but it was stated that such situations are uncommon and can be ignored. In this work we show that it is possible in cases to arrive at significantly better solutions by considering non-Hanan points during Steiner tree construction for two problems:
(a) the minmax problem, and

(b) the problem of achieving a specified delay at each sink node.
It should be pointed out that the problem (b) above can be transformed into the form of problem (a).

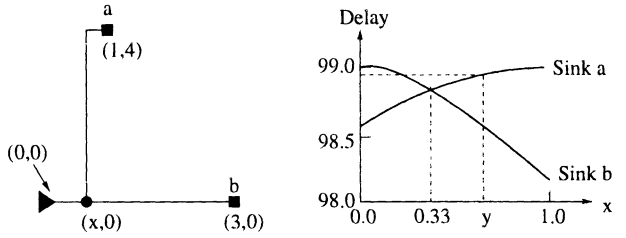


Figure 7: Illustrating the effects of non-Hanan Steiner points.

Example 1: We illustrate a simple example in Figure 7 showing that a non-Hanan node is required to minimize the maximum source-sink delay during tree construction. Consider a net with a source at (0,0) and two sinks, a and b, at (1,4) and (3,0), respectively. We assume, for simplicity, a unit resistance and a unit capacitance per unit length. The driver has a source resistance of 6, and the sinks a and b have load capacitances of 1 unit and 4.5 units, respectively. The delays are calculated here under the Elmore delay model, described in Section 2.2. The variation of the delay at each sink as the Steiner point x is moved from (0,0) to (1,0) is shown in Figure 7[1]. The maximum sink delay for the tree is minimized at $x = 0.33$[2].

This example illustrates that in real design problems, the timing requirements at different sinks are often contradictory and it is necessary to arrive at a solution where all of the sinks are considered together.

For the problem of achieving a specified timing constraint, it is also easy to show that the optimal solution may lie at a non-Hanan point. Any procedure that restricts of Steiner points to Hanan points alone would lead to a larger than optimal tree cost. Therefore, for the problems of achieving a set of specified sink delays, and of minimizing the maximum source-sink delay, the best Steiner points do not necessarily lie on the Hanan grid.

3.2 Problem Formulation

We now describe a procedure for building a minimum cost tree that satisfies the delay constraints at each sink. The procedure described in this section is

[1]The logic in [11] can be extended to show that a Steiner point to the right of (1,0) is suboptimal.

[2]We caution the reader not to be unduly swayed by the modest delay reductions in this simple example, it is possible to achieve more significant improvements on larger examples.

referred to as the Maximum delay Violation Elmore Routing Tree (MVERT) algorithm. Clearly, a positive value of the delay violation implies that the constraints could not be met. A large negative value of the violation, on the other hand, indicates the possibility of overdesign, and it is possible in some cases to reduce the cost of the Steiner tree by bringing the violation value to be closer to zero. This idea motivates the formal statement of the MVERT problem as follows:

Problem 3.1 *Given a signal net N with source v_0 and a set of sinks $V_{sink} = \{v_1, v_2, ..., v_n\}$ construct a Steiner routing tree $T(V, E)$ such that the total length of the net is minimized while the delay violation at each sink node is non-positive, i.e.,*

$$\begin{array}{ll} \textit{minimize} & \sum_{\forall e_k \in E} l_k \\ \textit{subject to} & t(v_i) \leq q(v_i) \forall v_i \in V_{sink} \end{array} \tag{2}$$

3.3 Method for Finding Optimal Non-Hanan Steiner Nodes

Since the restriction to a Hanan grid is no longer valid, the set of candidate Steiner points is infinite, and it is necessary to find an efficient method to identify the best Steiner points. We will introduce a method that utilizes the properties of the delay function to arrive at a simple and efficient method to overcome this challenge.

As defined in [11], a maximal segment is a set of contiguous edges either all vertical or all horizontal. The work of [11] shows that the Elmore delay at each sink is a concave function with respect to the location of a Steiner node when the Steiner is moved along a maximal segment. Although by definition, the orientation for a soft edge is not fixed, the concavity property continues to hold for a soft edge, and we can extend the philosophy of non-Hanan optimization to general edges including both solid and soft edges.

For a general form of a routing tree, shown in Figure 8, let us consider the process of obtaining an optimal connection between node v_k and edge e_{ij}. The dashed lines are other nodes and edges of this routing tree, and CC represents the closest connection point between v_k and e_{ij}. It can be easily seen that any connection that is downstream of CC cannot give an optimal solution [11]. More specifically, we wish to search for an optimal connection point within the bounding box defined by v_i and CC. Suppose we connect v_k to e_{ij} at point $v'(x', y')$. Let z be the Manhattan distance from v' to v_i, i.e., $z = |x' - x_i| + |y' - y_i|$. For convenience, we overload CC as its Manhattan distance to v_i. Similar to the work of [11], a delay function

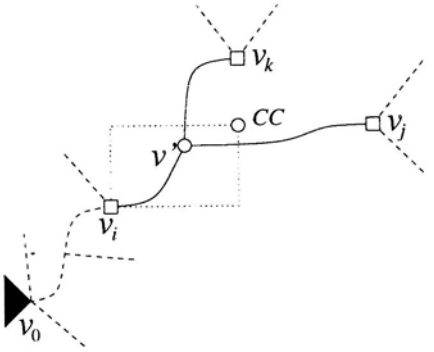


Figure 8: A general situation where node v_k is to be connected to an edge e_{ij}.

model with respect to connection location for soft edges under the Elmore delay is derived as follows.

If a node is not downstream of node v_i, its Elmore delay from source is as follows:

$$f_1 = R_d(C_t - cz) + \lambda_0 + \lambda_1(l_{ik} - z), \tag{3}$$

where λ_0 and λ_1 are constants. We use C_t to denote the total load capacitance seen from the last stage of driver if v_k is connected to v_i.

The Elmore delay from v_i to v' is given by:

$$f' = rcz(\frac{z}{2} + l_{ij} - z + l_{ik} - z) + rz(C_j + C_k). \tag{4}$$

From v' to any node in T_j which is the subtree rooted at v_j, the delay can be obtained as:

$$f_2 = r(l_{ij} - z)(\frac{c(l_{ij} - z)}{2} + C_j) + \lambda_2. \tag{5}$$

Similarly, the delay from v' to any node in T_k is:

$$f_3 = r(l_{ik} - z)(\frac{c(l_{ik} - z)}{2} + C_k) + \lambda_3. \tag{6}$$

Both λ_2 and λ_3 are constants. If a sink is in T_j, its Elmore delay is formed by the sum of f_1, f' and f_2. When a sink is in T_k, its Elmore delay is the sum of f_1, f' and f_3. If a sink is not downstream of v_i, its Elmore delay is simply f_1. In all these cases, the delay is either a linear or a quadratic function of z with non-positive coefficient for the second order term. Therefore, we can obtain the conclusion that delay for any sink is a concave function with respect to z, which is concluded as follows.

Theorem 3.1 *Under the Elmore delay model, the delay at any sink in the routing tree is a concave function with respect to z.*

Consider the set of constraints on the routing tree from Equation (2). Rewriting them in the form $t(v_i) - q(v_i) \leq 0$ for all sinks $v_i \in V_{sink}$, we see that the maximum violation must always be non-positive. Since each of the $t(v_i)$'s is a concave function of the connection point z by Theorem 3.1, and since any concave function shifted by a constant is a concave function, this implies that we must find a reconnection point z such that the maximum of the set of concave functions is non-positive. This is pictorially shown in Figure 9 for a net with four sinks, u, v, w, and y; they have the same timing specification q. The maximum violation function is shown by the darkened line. This piecewise concave function is composed of three concave *pieces.* Note that the graph shows that sink u is never critical in this case, for any value of z. The delay violation at each sink as a function of z is a concave function and the objective is to find the value of z that is closest to CC (corresponding to a minimal increase in the net length) that satisfies all constraints. In Figure 9, this point is found to be z^*. This point would, in general, be a non-Hanan point.

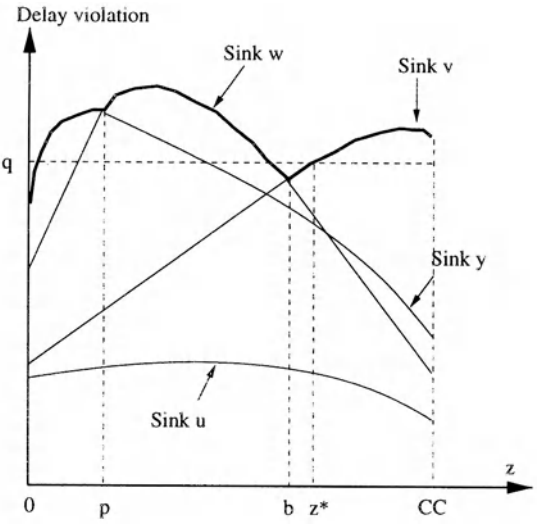


Figure 9: Finding the optimal value of z that satisfies the timing constraints.

In searching for z^*, we observe that it is possible to perform a search on the value of z from 0 to CC, while taking advantage of the fact that the value on each concave piece is minimized at its intersection with the

concave piece on either side (if such a piece exists), or at 0 or CC otherwise. In Figure 9, this translates to the fact that for the minmax problem, the only candidate solutions are 0, p, b and CC. This permits a reduction of the search space from the infinity of points between 0 and CC.

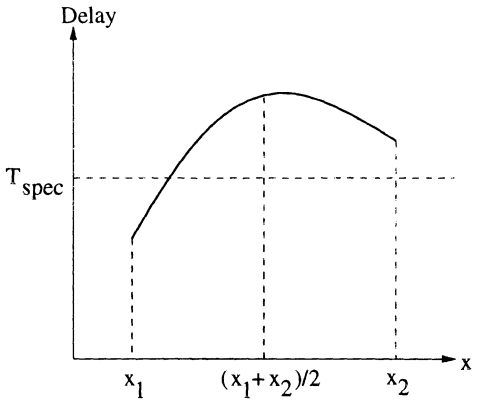


Figure 10: Using piecewise concavity to speed up the optimization procedure.

For the problem of meeting timing specifications at each sink, several pruning strategies are possible for the search. Consider a binary search on a concave piece with end points x_1 and x_2 ($x_1 < x_2$) with values $f(x_1)$ and $f(x_2)$, respectively. If $T_{spec} > f(x_1)$, $T_{spec} < f(x_2)$ and $T_{spec} < f(\frac{x_1+x_2}{2})$ as illustrated in Figure 10, then the search can completely eliminate the interval $[\frac{x_1+x_2}{2}, x_2]$. This follows from the fact that any concave function over an interval is concave over any continuous subinterval. By a symmetric argument, if $T_{spec} \geq f(\frac{x_1+x_2}{2})$, then the search is reduced to the interval $[\frac{x_1+x_2}{2}, x_2]$.

The pseudocode corresponding to this search is shown in Figure 11. The routing tree without subtree T_k is represented by $T \backslash T_k$. The efficiency of the search can be greatly enhanced by taking advantage of the piecewise concavity of the delay function. The search for z^* is between 0 and CC in a binary search fashion, and begins at CC. If the value of the delay violation at CC is negative, then we are done; otherwise we need to test at 0. We use CS to represent the critical sink that has the maximum delay violation $\Delta_{max} = \max\{t(v_i) - q(v_i), \forall v_i \in V_{sink}\}$. If the Δ_{max} is positive at both 0 and CC, and the critical sink at 0 is the same as at CC, then there is no solution satisfying timing constraints. In this case, we choose the one gives less delay violation between 0 and CC. A more complicated situation is when Δ_{max} at 0 is negative, or Δ_{max} is positive at 0 but the corresponding

Algorithm: FindOptimalConnection$(T_k, T\backslash T_k, e_{ij})$

Input: Subtree T_k rooted at sink v_k
Partial routing tree $T\backslash T_k$, edge $e_{ij} \in T\backslash T_k$

Output: Optimal connection between v_k and e_{ij}

1. Tentatively join v_k to CC, $\Delta_{rit} \leftarrow \Delta_{max}$,
   $CS_{rit} \leftarrow$ sink with $\Delta_{max}, S_{rit} \leftarrow (CC, \Delta_{rit}, CS_{rit})$
2. If $\Delta_{rit} \leq 0$, return CC
3. Tentatively join v_k to v_i
   $CS_{lft} \leftarrow$ sink with $\Delta_{max}, S_{lft} \leftarrow (v_i, \Delta_{max}, CS_{lft})$
4. Return $Search(S_{lft}, S_{rit})$

Function: $Search(S_{lft}, S_{rit})$

F1. If $\Delta_{rit} \leq 0$, return S_{rit}
F2. If ($\Delta_{lft} > 0$ and $CS_{lft} == CS_{rit}$) or $dist(v_{lft}, v_{rit}) <$ resolution
F3. If $\Delta_{lft} < \Delta_{rit}$, return S_{lft}
F4. Else return S_{rit}
F5. $v_{mid} \leftarrow ((x_{lft} + x_{rit})/2, (y_{lft} + y_{rit})/2)$
F6. Join v_k to e_{ij} at $v_{mid}, \Delta_{mid} \leftarrow \Delta_{max}$
$CS_{mid} \leftarrow$ sink with $\Delta_{max}, S_{mid} \leftarrow (v_{mid}, \Delta_{mid}, CS_{mid})$
F7. If $\Delta_{mid} \leq 0$, return $Search(S_{mid}, S_{rit})$
F8. $S_r \leftarrow Search(S_{mid}, S_{rit})$
F9. If $\Delta_r \leq 0$, return S_r
F10. $S_l \leftarrow Search(S_{lft}, S_{mid})$
F11. If $\Delta_l < \Delta_r$, return S_l
F12. Else return S_r

Figure 11: Algorithm for finding an optimal connection point between a sink and an edge.

critical sink is different from that at CC. Then, the search proceeds as a quasi-binary search, as shown in the function $Search(S_{lft}, S_{rit})$ in Figure 11. The notation S indicates a solution which is a triple of (connection node, Δ_{max}, critical sink), and S_{lft} and S_{rit} imply the solutions at the left and right end of the search interval. If the size of the interval is less than a user specified resolution, the search terminates(line F2-F4 in Figure 11). If the connection at the middle point of the interval yields a non-negative Δ_{max}, the search continues only on the right half of the interval (line F7 in Figure 11); otherwise, the left half may be searched as well(line F8-F12 in Figure 11).

3.4 Non-Hanan Optimization Flow

The MVERT algorithm is divided into two phases: (I). The initial tree construction phase, where an initial tree is heuristically built to minimize delay. (II). The cost-improvement phase, where the tree is iteratively refined to reduce its cost while ensuring that it meets all timing specifications.

The tree construction in Phase I is similar to the SERT construction procedure in [11]. The essential idea of the SERT method is based on building a greedy Steiner tree using an approach similar to Prim's algorithm. Starting with a trivial tree T consisting of only the source v_0, the tree is iteratively built by joining a sink v_k outside the tree to an edge (or the source) in the tree so as to yield a tree with the minimum Elmore delay. The iterations continue until all sinks have been included in the tree.

The initial tree constructed above considers only Hanan grid points as candidate Steiner points. Therefore, it attempts to connect each point either to the closest connection (CC) or the upstream end of an edge, or the source directly. If the delay associated with a CC connection were larger than the delay associated with a connection to the upstream end, then the algorithm will not choose connection at CC. However, due to the interactions between paths the MVERT solution may lie at a different (and possibly non-Hanan) point, and the connection to the upstream end of an edge may result in a larger net length than is necessary. Therefore, we examine the tree constructed in Phase 1 and move node connections from the upstream end of an edge towards CC in a bid to reduce the tree length while ensuring that all timing constraints are satisfied. The idea is illustrated in Example 1 (Figure 7) for the constraint of 98.8 units, where we see that a connection to (y,0) is preferable to a connection to (0.33,0).

The pseudo code for the non-Hanan optimization is shown in Figure 12. We sort all the sinks in a descending order of distance from the source. For each sink v_k we disjoin it and its downstream subtree T_k and reconnect it back. The reconnection is tested to each edge e_{ij} in the remaining tree $T \backslash T_k$. The optimal connection point is found through the procedure described in Figure 11. Finally, we choose the edge providing the largest improvement to join v_k and T_k. If a routing tree has a positive Δ_{max}, an improvement refers to a less Δ_{max}; otherwise, a wire length reduction implies an improvement. The experimental results in [7] show that non-Hanan optimization can provide about $5\% - 40\%$ timing-constrained wire length reduction over the SERT algorithm.

Algorithm: Non-Hanan_Optimization(T)

Input: Routing tree $T(V, E)$
Output: Optimized routing tree T'

1. $T' = T$
2. Sort all the sinks in descending order of distance to source
3. For each $v_k \in V_{sink}$
4. Disjoin v_k and its subtree T_k from T
5. For each edge $e_{ij} \in T \backslash T_k$
6. Reconnect v_k to e_{ij} at FindOptimalConnection($T_k, T \backslash T_k, e_{ij}$)
7. If $\exists$ improvement compared to T'
8. $T' = T$
9. Return T'

Figure 12: The non-Hanan optimization algorithm.

3.5 Complexity Analysis

As shown in [11], the computational complexity in Phase I is $O(n^4)$, where n is the number of sinks. In Phase II, each sink is processed precisely once, in routine showing in Figure 12. The cost involved in processing a sink v_k is in finding a reconnection point. The connection is tested for at most $O(n)$ edges in the remaining tree of $T \backslash T_k$. The complexity of finding optimal connection between a node and an edge is determined by the number of iterations K in the procedure of Figure 11. In each iteration, $O(n)$ computation time is spent on Elmore delay calculation. Thus the complexity in Phase II is $O(n^3 \cdot K)$. If the maximum edge length in tree T is l_{max} and the resolution in procedure of Figure 11 is ϵ, then the number of iterations is bounded by $\min\{n, \frac{l_{max}}{\epsilon}\}$, since there are at most $O(n)$ pieces in the piecewise concave Δ_{max} function. Therefore, the complexity of Phase II is is $O(n^3 \cdot \min\{n, \frac{l_{max}}{\epsilon}\})$, and the complexity for MVERT algorithm is $O(n^4)$.

4 Enhanced Non-Hanan Optimization for Timing Critical Nets

4.1 The Motivation for Using Higher Order AWE Model

As interconnect wires become increasingly thinner and longer, the interconnect resistance may overshadow the driver resistance. Consequently, the downstream capacitance is shielded to the driver resistance by the intercon-

nect resistance. This effect is called resistive shielding [13]. The Elmore delay does not correctly take the resistive shielding effect into account and tends to overestimate the delay. This error can be remarkably large, especially for the stub situation (i.e., when a sink that is close to the source co-exists with a much longer wire), where the Elmore delay can be several times larger than the actual delay.

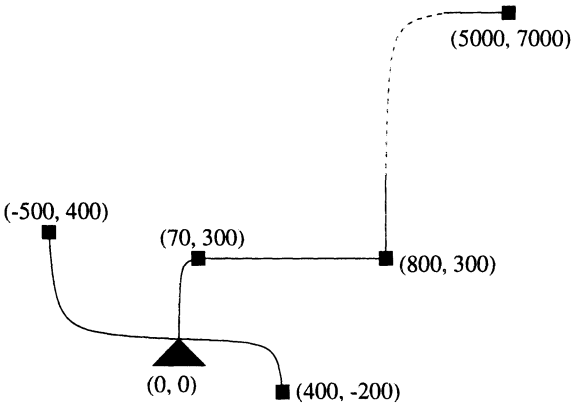


Figure 13: A routing tree on which Elmore delay gives large errors.

Table 1: A comparison of the Elmore and the 4th order AWE delays with SPICE

Dist.	SPICE	Elmore	Error	4th AWE	Error
370	13.6	52.5	286%	12.8	-6%
600	9.5	39.8	319%	8.9	-6%
900	10.7	40.5	279%	10.5	-2%
1100	26.2	77.4	195%	25.5	-3%
12000	283.2	257.5	-9%	282.4	-0.3%

Table 1 shows an example of a net with five sinks to illustrate the inaccuracy of the Elmore delay. The routing topology of this net is illustrated in Figure 13. The load capacitance is the same for each sink. The delays at all sinks are computed using the Elmore formula, fourth order AWE and a SPICE transmission line model, and the percentage errors relative to SPICE are calculated. The Manhattan distance from each sink to the source are also listed for reference. We can see that the error of Elmore delay can be over 300% and the delay from fourth AWE is clearly superior. In fact, as the minimum feature size shrinks, this trend will become more and more severe.

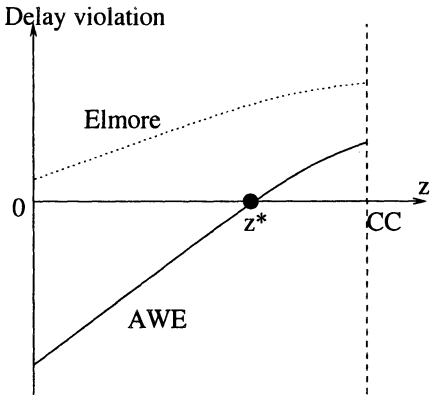


Figure 14: An example where using the Elmore delay and a higher order AWE delay may result in a different connection choice.

To see how this will affect non-Hanan routing, consider the graph in Figure 14. The graph plots the delay violation function against the location of the connection point, z. The dotted curve indicates the Elmore delay while the solid curve represents the fourth order AWE result. The solution corresponds to the point closest to CC where the delay violation function is negative or zero. For the Elmore delay, which overestimates the delay near the source, no solution is found, whereas an actual solution exists and corresponds to z^*.

On the other hand, we have observed that the Elmore model tends to under-estimate delay at sinks far from the source[3]. This may lead to the opposite error, as can be seen in the last row of Table 1. This under-estimation may result in over-reduction of cost while the timing constraints have not been satisfied yet. On the whole, a higher order model is greatly superior to the Elmore model in handling non-Hanan points.

The conclusion from Section 3 on non-Hanan optimization is also valid under a higher order AWE model according to the experimental results shown in Table 2 and 3. The leftmost column in each table lists the number of sinks in each net. The technology parameters are same as those in [9]. The experiment starts by constructing routing trees for each net through SART, which is same as SERT [11] except that the Elmore delay model is replaced by a fourth order AWE model. The experimental results from this

[3]The Elmore delay is theoretically proven to be an upper bound on the delay of an RC network in [14]. However, in practice, greater accuracies are obtainable by multiplying the Elmore delay formula of [15] by a factor of $\ln 2$, and we refer this quantity as the *Elmore delay* in our discussion, and this may be either optimistic or pessimistic.

Table 2: Comparisons between using and without using Hanan nodes under fourth order AWE model on $.18\mu m$ IC technology.

	SART		Hanan		NonHanan	
n	Δ_{max}	W	Δ_{max}	W	Δ_{max}	W
5	38.4	226	17.4	258	-9.4	226
5	10.1	204	-1.0	209	-24.9	188
5	42.9	288	-29.2	290	-6.2	279
10	17.3	388	7.3	381	-3.5	354
10	36.9	417	12.4	383	-3.1	396
10	52.4	502	7.5	526	-7.7	491
15	27.1	570	-5.2	539	-4.7	488
15	81.5	583	-3.1	508	-7.1	456
20	97.7	604	-5.0	612	-25.4	555
20	7.1	671	-2.8	666	-1.4	616
Ave	41.1	445	-0.2	437	-9.3	405

step are provided in column 2 and 3 in Table 2 and 3. The total wire length is denoted as W in unit of $100\mu m$ and the maximum delay violation for each net is represented as Δ_{max} in ps. Then, the optimization scheme in Figure 12 are performed on the SART trees also under a fourth order AWE model. We restrict the connection point in line 6 of Figure 12 to be only Hanan point in one variant of this optimization whose results are shown in column 4 and 5 in both tables. The results from original non-Hanan optimization are in the rightmost two columns. We can see that routing solution using only Hanan points sometimes results in positive delay violations, and these delay violations may be eliminated through using non-Hanan points. Moreover, using non-Hanan points can yield more wire length reductions.

In the computation of fourth order AWE delay, we first use the RICE algorithm [16] to obtain the moments. We solve the denominator of Padé approximation result, which is a fourth order polynomial, using a closed-form formula to obtain the poles. After an inverse Laplace transformation, the time-domain exponential functions are expanded about the Elmore delay to fourth order Taylor series polynomials. A closed-form solution to a fourth order polynomial exists and may be used to calculate the delay value. Since the Elmore delay may be far off from correct value, sometimes the expansion about Elmore delay may still cause significant error, though it is much smaller than the error from the Elmore delay. We restrict such error by another iteration with expansion about the result from the first iteration. This process is iterated until convergence, and we found that we

Table 3: Comparisons between using and without using Hanan nodes under fourth order AWE model on MCM technology.

	SART		Hanan		NonHanan	
n	Δ_{max}	W	Δ_{max}	W	Δ_{max}	W
5	20.0	585	-17.9	543	-9.8	502
5	68.9	428	-2.3	532	-1.6	490
5	13.7	499	6.7	480	-19.5	472
10	93.0	803	14.3	760	-8.3	784
10	25.1	819	21.2	782	-1.6	662
10	18.2	845	-2.1	924	-2.3	806
15	42.4	1258	-4.0	1221	-14.2	1192
15	48.3	1110	-54.5	1119	-35.8	1004
20	64.8	1518	11.2	1469	-11.2	1410
20	268.1	1473	-0.5	1445	-12.0	1286
Ave	66.2	934	-2.8	927	-11.6	861

always converged within three iterations. This method is related to the Newton-Raphson root-finding method: the Newton-Raphson method uses a first order Taylor series in each iteration, and our method uses a fourth order expansion instead.

The reason that we choose fourth order instead of a second or third order model is that a second order yields less accuracy and for many examples that we tried, and we found that the third order model induces positive poles more often. The computation overhead for models with order greater than four is large, since there is no closed form solution for equations with order beyond four. The additional computation cost of fourth order AWE as compared to a second order model is minor.

4.2 Integrated Buffer Insertion and Non-Hanan Optimization

We will present the integrated Buffer Insertion and Non-Hanan Optimization (BINO) algorithm for the timing critical nets. The motivation for combining buffer insertion with non-Hanan optimization can be illustrated by the example in Figure 15. In order to reduce wire cost, it is desired to move the connection point as close to CC as possible, i.e., to maximize z. However, the value of z may be capped by the constraint of non-positive delay violation as illustrated in Figure 15(a). The utility of buffer insertion is to relax this timing constraint, if possible, so as to achieve further wire cost reduction as in Figure 15(b).

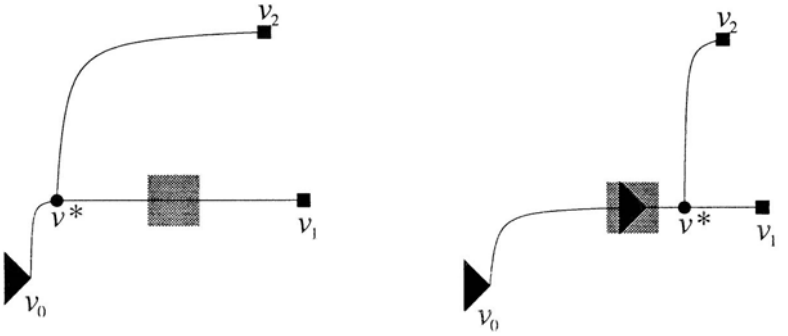


Figure 15: An example that buffer insertion can reduce wire cost further in non-Hanan optimization.

This algorithm is applied in a post-placement scenario where buffer insertion is possible, but it is preferable to do so in regions that are left unoccupied by any cells, so as not to disturb the placement. This method is also applicable to MCM technology, where a buffer location is desired to be within a chip and close to its chip bond pads, because it is not cost-effective to insert a buffer either on the substrate between chips or within a chip but far from any bond pad. The input to BINO then includes a set of pre-defined available buffer spaces scattered in the routing region. These buffer spaces are represented by small squares, as demonstrated by the dark grey areas b_1 and b_2 in Figure 16 (a). It is assumed that only one buffer can be inserted in each space and the center of the buffer must lie within the square. Larger buffer spaces can easily be expressed as a union of small spaces.

Intuitively, a buffer space is considered for buffer insertion only when a routing path passes through it, since no extra wire cost is incurred under this condition. However, even if no path passes through a buffer space, it may be worthwhile for the wire to make small detour to increase the possibility of exploiting a buffer space. Based on this idea, we define a territory box for an edge as follows:

Definition 4.1 (territory box) *For an edge e_{ij}, its territory box is a rectangle specified by lower-left corner point (x_{min}, y_{min}) and upper-right corner point (x_{max}, y_{max}), such that:*

$$x_{min} = \min(x_i, x_j) - \phi,$$
$$y_{min} = \min(y_i, y_j) - \phi,$$
$$x_{max} = \max(x_i, x_j) + \phi,$$
$$y_{max} = \max(y_i, y_j) + \phi,$$

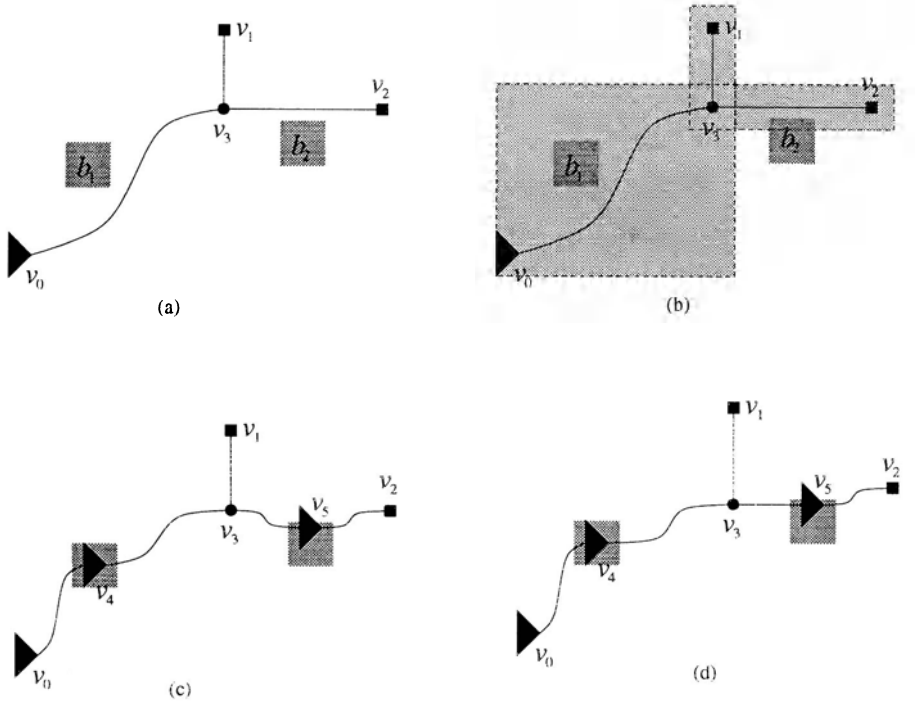


Figure 16: Buffer spaces, the territory box and their applications in buffer insertion.

where ϕ is a small amount of offset.

The idea of a territory box is demonstrated by the light grey regions in Figure 16(b). Note that the territory box for the soft edge e_{03} is larger than for any solid edges between v_0 and v_3. The rule that we will follow is as follows: *a buffer space is considered for buffer insertion in an edge only when there is an overlap between this buffer space and the territory box of this edge.* In the example of Figure 16, buffer space b_1 overlaps with the territory box of edge e_{03} and b_2 overlaps with the territory box of e_{32}; therefore, we can insert buffers v_4 and v_5 as in Figure 16(c). After the non-Hanan optimization following the buffer insertion, the wire slack in Figure 16(c) may be removed and the tree shown in Figure 16(d) may be obtained. This example shows that the use of soft edges can greatly increase the possibility of overlap as compared to using predetermined L-shaped connection composed of two solid edges.

We consider both inverting and non-inverting type buffers in our work. The inverting type buffer is simply an inverter and the non-inverting type

buffer is composed by a pair of cascaded inverters. The inverter model is same as the driver model in Section 2 and has a medium driver size.

We use $\Delta(v_i)$ to represent the delay violation at sink v_i. The gate and drain capacitance of an inverting buffer are denoted as C_{gb} and C_{db}. The total wire length is represented as W and γ is the weighting factor for the wire cost. The simultaneous buffer insertion and non-Hanan optimization problem is to minimize a weighted sum of buffer and wire cost subject to timing constraints. This is formulated as follows.

Problem 4.1 *Given a source v_0, a set of sinks $V_{sink} = \{v_1, v_2 ... v_n\}$, timing specifications $Q = \{q_1, q_2, ..., q_n\}$ for all sinks and a set of available buffer spaces $P = \{p_1, p_2, ..., p_m\}$, construct a Steiner routing tree and choose a subset $B_{iv} \subseteq P$ and $B_{ni} \subseteq P$ on which inverting and non-inverting buffers are inserted, respectively, such that the following is solved:*

$$\begin{array}{cc} \textit{minimize} & \gamma cW + (1-\gamma)(C_{gb} + C_{db})(|B_{iv}| + 2|B_{ni}|) \\ \textit{subject to:} & \max_{v_i \in V_{sink}} \Delta(v_i) \leq 0 \\ \textit{for a specific } \gamma & 0 \leq \gamma \leq 1 \end{array} \tag{7}$$

The purpose of including c (wire capacitance per unit length), C_{gb} and C_{db} in the objective function is to normalize the wire and the buffer cost into comparable quantities.

The algorithm of BINO consists of two phases. Phase I is the routing tree construction process called SART (Steiner AWE Routing Tree), and is similar to SERT [11] except that the Elmore model is replaced by a fourth order AWE model and soft edges are employed.

In Phase II of BINO, the non-Hanan optimization framework in Figure 12 is embedded in a greedy buffer insertion scheme illustrated by Figure 17. On each buffer space, we insert a buffer tentatively and conduct non-Hanan optimization. After all of the buffer spaces have been tested, the solution that can provide the largest improvement is chosen as the final decision. This process is repeated iteratively until there is no improvement or no buffer space left. The optimal solution of assigning inverting or non-inverting type to each buffer (line 6 in Figure 17) can be achieved through dynamic programming.

Since we only insert one buffer in each iteration, the ability to obtain an optimal buffer insertion solution is hindered, as shown by the single-sink example in Figure 18. It is well known that optimal buffer locations often distribute evenly along an interconnect path [17]. Therefore, for the net in Figure 18, the optimal solution may be as shown in Figure 18(d). If we insert only one buffer in an iteration, the first iteration is likely to result

Algorithm: BINO_IterativeBufferInsertion
Input: SART $T(V, E)$, a set of buffer spaces P
Output: Buffered and non-Hanan optimized routing tree T
1. While $P \neq \emptyset$ and $\exists$ improvement
2. For each $p \in P$
3. For each edge $e_{ij} \in E$
4. If p overlaps with the territory box of e_{ij}
5. Insert a buffer into e_{ij} at p tentatively
6. Assign inverting/non-inverting type $\forall$ buffers $\in T$
7. Perform non-Hanan optimization for T (Figure 12)
8. Insert buffer at p_{best}, which gives the largest improvement
9. $P \leftarrow P - p_{best}$

Figure 17: BINO, iterative buffer insertion algorithm.

in the scenario shown in Figure 18(b) and the optimal solution cannot be reached.

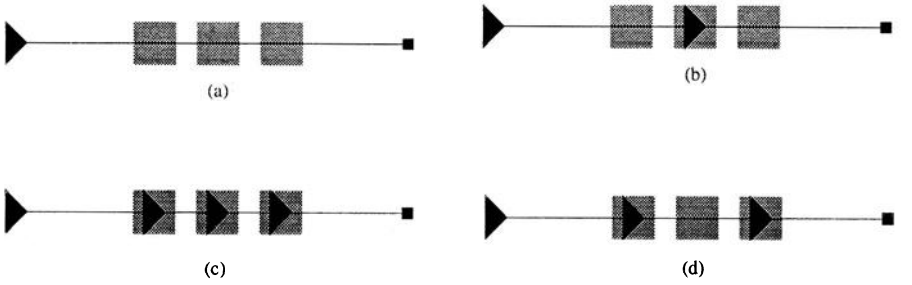


Figure 18: Iterative buffer insertion vs. iterative buffer deletion.

In order to alleviate the above difficulty, we supplement the method with an iterative buffer deletion procedure using a method similar to [18], that is described in Figure 19. In this scheme, we first insert buffers at all spaces that overlap with any edges. Then we delete one buffer in each iteration in a greedy fashion similar to iterative buffer insertion. Since this proceeds in the opposite direction as compared to the iterative buffer insertion, it plays a complementary role. For the example in Figure 18, the iterative buffer deletion starts with (c) and can naturally result in the optimal solution in (d). On the other hand, if the optimal solution is (b), iterative buffer deletion is worse than iterative buffer insertion.

In our work, we perform both iterative buffer insertion and iterative buffer deletion independently for a net and choose the better of the two

Algorithm: BINO_IterativeBufferDeletion
Input: SART $T(V, E)$, a set of buffer spaces P
Output: Buffered and non-Hanan optimized routing tree T
1. $B \leftarrow \emptyset$
2. For each $p \in P$
3. For each edge $e_{ij} \in E$
4. If p overlaps with the territory box of e_{ij}
5. Insert buffer b into e_{ij} at p
6. $B \leftarrow B \cup b$
7. Assign inverting/non-inverting type $\forall b \in B$
8. While $B \neq \emptyset$ and $\exists$ improvement.
9. For each buffer $b \in B$
10. Remove b from T tentatively
11. Assign inverting/non-inverting type $\forall b \in B$
12. Perform non-Hanan optimization for T (Figure 12)
13. Remove buffer b_{best}, which gives the largest improvement
14. $B \leftarrow B - b_{best}$

Figure 19: BINO, iterative buffer deletion algorithm.

results.

4.3 Simultaneous Non-Hanan Optimization and Driver Sizing

The task of delay optimization of a circuit consists of appropriately optimizing the gates and optimizing the interconnect wires. The gates may be optimized through actions such as mapping the circuit to a set of complex gates (possibly from a library) and/or sizing the gates to achieve the requisite driving power. For optimal circuit design, the cost of the optimization must be borne equally by the gates and by the interconnects in an interconnect-dominated environment. The driver sizing problem is to choose optimal number of driver stages h and the proper size for each driver. We choose the ratio of driver size at one stage to its previous stage to be uniform and refer to it as the stage ratio ρ. In this section, we introduce the algorithm of simultaneous non-Hanan optimization and driver sizing, which is called FAR-DS (Full-plane AWE Routing with Driver Sizing).

The objective of FAR-DS is to minimize the cost of the routing tree, subject to a timing constraint at each sink. In contrast with MVERT, we

extend the cost here to include both wire cost and driver cost, i.e., we perform topology optimization and driver sizing simultaneously. The rationale behind this is to permit the driver to share the task of delay optimization with the interconnect by sizing it, thereby obtaining a better result than optimizing the driver size and interconnect topology separately. We formally state the problem formulation as follows.

Problem 4.2 *Given a source v_0, a set of sinks $V_{sink} = \{v_1, v_2 ... v_n\}$, timing specifications $Q = \{q_1, q_2, ... q_n\}$ for all sinks, and stage ratio bound ρ_{max}, construct a Steiner routing tree and find ρ, h such that:*

$$\begin{array}{cc} \textit{minimize} & \gamma cW + (1-\gamma)(C_g + C_d)\sum_{j=1}^{h} \rho^j \\ \textit{subject to:} & \max_{v_i \in V_{sink}} \Delta(v_i) \leq 0 \\ \textit{and} & 1 \leq \rho \leq \rho_{max}. \end{array} \tag{8}$$

The second term in the objective function is from the total driver capacitance. The objective function can be interpreted as a minimization of the total wire length and total driver capacitance. The parameter γ is a user-specified weighting factor.

For a general connection of a node and its downstream subtree to a partial tree, as illustrated in Figure 8, where a node v_k is to be connected to an edge e_{ij}, we investigate the properties of the delay violation function with respect to z and ρ in a two dimensional space. The delay from the cascaded drivers is given by:

$$T_D = hR_0(C_d + \rho C_g) \tag{9}$$

We can combine the interconnect delay discussed in Section 3 with T_D to obtain a general form of the delay violation of any sink $\Delta(v_a)$ as a function of the connection position z and ρ, under the Elmore model as:

$$\Delta(v_a) = f(z, \rho) = -a_2 rcz^2 + \frac{R_0(C_t - cz)}{\rho^h} + a_1 z + R_0 C_g h\rho + a_0 \tag{10}$$

where

$$a_2 = 0 \text{ or } 1, \quad 0 \leq z \leq CC < \frac{C_t}{c}, \quad 1 \leq \rho \leq \rho_{max}, \tag{11}$$

with a_0 and a_1 being constants. The parameter C_t is the total load capacitance seen by the driver in the last stage when v_k is connected to v_i directly.

When ρ is fixed, $\Delta(v_a) = f(z)$ is a quadratic function of z and the coefficient of the second order term is non-positive. Therefore we can obtain the following result:

Property 4.1 $\Delta(v_a) = f(z, \rho)$ *is a concave function for a constant value of* ρ.

If we keep z constant, there are also properties that will help the search for the optimal solution. These properties can be found by investigating the partial derivatives of $\Delta(v_a)$ with respect to ρ as follows:

$$\frac{\partial \Delta(v_a)}{\partial \rho} = -R_0(C_t - cz)h\rho^{-h-1} + R_0 C_g h \tag{12}$$

$$\frac{\partial^2 \Delta(v_a)}{\partial^2 \rho} = R_0(C_t - cz)h(h+1)\rho^{-h-2} \tag{13}$$

Since $C_t > cz$, $\frac{\partial^2 \Delta(v_a)}{\partial^2 \rho} > 0$ is always true, thus we have the following property:

Property 4.2 $\Delta(v_a) = f(z, \rho)$ *is convex function for a constant value of* z.

If we let $\frac{\partial \Delta(v_a)}{\partial \rho} = 0$, we can obtain a curve defined as follows:

$$\rho = \sqrt[h+1]{\frac{C_t - cz}{C_g}} \tag{14}$$

Property 4.3 $f(z, \rho)$ *has minimum value along the curve defined by equation (14).*

This property is especially useful in solution search, since it predicts the bottom of the valley shaped delay violation function surface in the two-dimensional space of z and ρ. One observation is that the curve in equation (14) is independent of which sink is considered, i.e., equation (14) defines the bottom of valley for the delay violation functions of *all* the sinks. We call the curve defined by equation (14) the *valley curve* for delay violations.

In equation (14), when z is at CC, the numerator reaches the minimum and becomes the total load capacitance seen by the driver in the last stage when v_k is connected to CC. Obviously, this total load capacitance is always greater than the minimum gate capacitance, C_g, of a driver. This fact provides the following property:

Property 4.4 *If* $0 \leq z \leq CC$, *then* $\rho = \sqrt[h+1]{\frac{C_t - cz}{C_g}} > 1$.

If we substitute equation (14) into equation (10), we can obtain another important conclusion:

Property 4.5 $\Delta(v_a) = f(z, \rho)$ *is a concave function of* z *along the curve defined by equation (14).*

This valley curve also sets a border for different monotone properties with respect to ρ as follows:

Property 4.6 *For a specific* z, $f(z, \rho)$ *is a monotone decreasing function of* ρ *when* $\rho \leq \sqrt[h+1]{\frac{C_t - cz}{C_g}}$.

These properties are derived from Elmore delays. Though the Elmore delay may have large errors for specific points, its qualitative fidelity is still true [11] and can serve as good strategic guide. Our experimental results also support this assertion.

Same as BINO, the algorithm of FAR-DS also includes two phases with Phase I being SART. In Phase II of FAR-DS, a two-dimensional search replaces the role of the quasi-binary-search in MVERT, which is line 6 in Figure 12, to find an optimal connection point and driver size simultaneously.

When we reconnect a node v_k to an edge e_{ij}, we look for a 3-tuple (z, ρ, h) such that the objective function of Problem 4.2 is minimized while the delay violations for all sinks are non-positive. We vary h between 1 and h_{max} and search an optimal (z, ρ) pair in a two dimensional plane for a fixed h value.

For this case, $cW = C_t - cz$ and the objective in Problem 4.2 can be translated to:

$$\begin{array}{rc} \text{minimize} & g = -\gamma cz + (1-\gamma)(C_g + C_d)\sum_{j=1}^{h} \rho^j \\ \text{subject to:} & \max_{v_i \in V_{sink}} \Delta(v_i) \leq 0 \\ \text{and} & 0 \leq z \leq CC,\ 1 \leq \rho \leq \rho_{max}. \end{array} \tag{15}$$

For a specific value of g, the objective function above corresponds a curve in the (z, ρ) plane, as the objective curves shown in Figure 20. The objective (15) can be interpreted as to find a point in (z, ρ) plane such that the constraints in (15) are satisfied at this point and the point is on a objective curve as low as possible.

We will illustrate the optimal solution search scheme through the Figure 20. The solution search can be restricted within the rectangle bounded by $0 \leq z \leq CC$ and $1 \leq \rho \leq \rho_{max}$. Consider the valley curve defined by

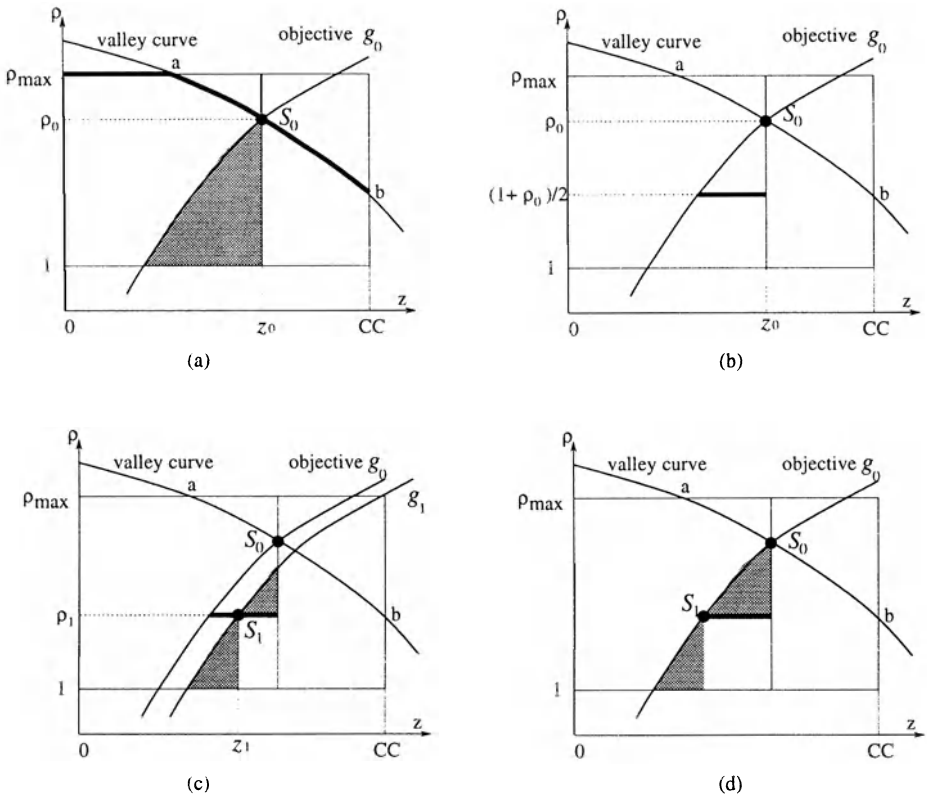


Figure 20: Solution search scheme for FAR-DS.

equation (14). This curve is always above $\rho = 1$ in the interval $0 \leq z \leq CC$, according to Property 4.4 . One common scenario is that this valley curve intersects with upper border of the rectangle at a point a and with the right border at b, as in Figure 20(a). From Property 4.3 and Property 4.6, we can say that in the rectangle defined above, $\Delta(v_i)$ reaches its minimum on the segment $\rho = \rho_{max}$ to the left of a, and on valley curve specified by equation (14) to the right of a. These two segments can be integrated into a single function:

$$\rho = \min(\rho_{max}, \sqrt[h+1]{\frac{C_t - cz}{C_g}}), 0 \leq z \leq CC \tag{16}$$

which is the thickened line in Figure 20 (a). Note that equation (16) is valid even when the valley curve does not intersect the rectangle, or if the set of points on the segment to the left of point a is empty. This function provides us with a convenient way to check for the existence of a solution within the

rectangle. From Property 4.3 and Property 4.6, if no solution that satisfies all constraints exists on the curve defined by equation (16), then we can say that no solution exists within the rectangle. According to Property 4.1 and Property 4.5, $\Delta(v_i)$ is a concave function on the curve (16), both to the left and to the right of point a. Thus, we can apply the quasi-binary-search technique in Section 3.3 to search for the rightmost solution on this curve that satisfies all constraints. If such a solution exists, we call it the zero order solution, designated as $S_0(z_0, \rho_0)$ in Figure 20(a).

After the zero order solution has been found, the region can be further refined to search for the optimal solution. This is demonstrated in the shaded region in Figure 20(a). The region $z > z_0$ can be excluded, since no feasible point exists on the valley curve in this region. An objective function curve is drawn through S_0, which satisfies:

$$g_0 = -\gamma c z_0 + (1-\gamma)(C_g + C_d)\sum_{j=1}^{h} \rho_0^j \tag{17}$$

We can eliminate the region that lies above this curve, because the value of g at all points above this line exceeds g_0. The remainder of the search space is the sector confined by the objective function curve defined by (17), by $z = z_0$ and by $\rho = 1$, which is indicated by the shaded region in Figure 20(a).

The search within this sector also proceeds in a binary search fashion, by starting from the middle segment defined by $\rho_1 = (1+\rho_0)/2$, which is the thickened segment in Figure 20(b). On this segment, Property 4.1 holds and a quasi-binary-search can again be applied to obtain the rightmost solution on it, namely, $S_1(z_1, \rho_1)$; we refer to this as the first order solution. After the first order solution has been found, the previously described solution refinement technique can be used to obtain two new smaller sectors shown by the shaded regions in Figure 20(c) where the optimal solution will be searched. Even if there is no solution on this segment, the search region can be refined to the two sectors like in (d). We call this solution search scheme as valley-guided search (V-search), and describe it in Figure 21.

The above is the method to search optimal (z, ρ) for a specific h. The optimal h is found by a sweep from $h = 1$ to $h = h_{max}$ and the above search is carried out for each h value. The value of h_{max} is given by [19]:

$$h_{max} = \left\lceil \frac{\ln(C_t/C_g)}{\ln \rho^*} \right\rceil, \tag{18}$$

Algorithm: FAR-DS_ReconnectVSearch

Input: Routing tree $T\backslash T_k$, subtree T_k, node v_k, edge e_{ij}
Output: Optimal connection between v_k and e_{ij}, ρ and h

1. For $h = 1$; $h \leq h_{max}$; h++;
2. Search solution along valley curve defined by equation(16)
3. If no solution found, return
4. $S_0(z_0, \rho_0) \leftarrow$ rightmost solution
5. SearchSector($S_0, 1$)

Function: SearchSector(S_{top}, ρ_{base})

F1. Obtain curve $g_{top} = -\gamma c z_{top} + (1-\gamma)(C_g + C_d) \sum_{j=1}^{h} \rho_{top}^j$
F2. $\rho_{mid} = (\rho_{top} + \rho_{base})/2$
F3. $S_{mid}(z_{mid}, \rho_{mid}) \leftarrow$ intersection between curve g_{top} and $\rho = \rho_{mid}$
F4. Search solution along $\rho = \rho_{mid}$ between z_{mid} and z_{top}
F5. If no solution found
F6. If $\rho_{mid} - \rho_{base} < resolution$, return no solution
F7. Return $SearchSector(S_{top}, \rho_{mid})$ and $SearchSector(S_{mid}, \rho_{base})$
F8. Else
F9. If $\rho_{mid} - \rho_{base} < resolution$, return rightmost solution
F10. $S_{mid}(z_{mid}, \rho_{mid}) \leftarrow$ rightmost solution
F11. Obtain curve $g_{mid} = -\gamma c z_{mid} + (1-\gamma)(C_g + C_d) \sum_{j=1}^{h} \rho_{mid}^j$
F12. $S_{top}(z_{top}, \rho_{top}) \leftarrow$ intersection between curve g_{mid} and $z = z_{top}$
F13. Return $SearchSector(S_{top}, \rho_{mid})$ and $SearchSector(S_{mid}, \rho_{base})$

Figure 21: FAR-DS, reconnection and driver sizing in valley-guided search.

$$\ln \rho^* = 1 + \frac{C_d}{C_g \rho^*}. \tag{19}$$

Since the use of valley curve increases the dependency of the solution on the Elmore delay model, and we use a higher order AWE model to evaluate the delays for every sink in our algorithm, it is possible that the discrepancy between Elmore model prediction and the actual AWE evaluation may give rise to a suboptimal solution.

We suggest an alternative search method called the iterative search (I-search) scheme that does not depend on Elmore model quantitatively and illustrate it in Figure 22. In this method, we begin with an initial ρ and perform non-Hanan optimization to obtain an optimal z for this value of ρ. Next, this z is fixed and an optimal ρ is searched and so on. This process

Algorithm: FAR-DS_ReconnectISearch	
Input:	Routing tree $T \backslash T_k$, subtree T_k, node v_k, edge e_{ij}
Output:	Optimal connection between v_k and e_{ij}, ρ and h
1.	For $h = 1$; $h \leq h_{max}$; h++
2.	$\rho \leftarrow$ initial guess
3.	While $\exists$ improvement
4.	Search z_{best} which gives best improvement while ρ is fixed
5.	$z \leftarrow z_{best}$
6.	Search ρ_{best} which gives best improvement while z is fixed
7.	$\rho \leftarrow \rho_{best}$

Figure 22: FAR-DS, reconnection and driver sizing in iterative search.

is repeated until there is no further improvement. From Property 4.2, we know that the delay violation function $\Delta(v_i)$ is a convex function along ρ direction, thus, we cannot apply the quasi-binary-search suggested in Section 3.3 along ρ direction. We perform the search in a manner between binary search and linear search. If the maximum delay violation is non-positive for a specific value of ρ, we continue to search a better solution at a smaller ρ value, otherwise, we must search at both larger and smaller values.

4.4 Complexity Analysis

From Section 3.5, the computation cost for MVERT is $O(n^4)$. Although we use the fourth order AWE instead of Elmore in BINO, as the number of iterations is fixed, the complexity for each delay calculation remains $O(n)$. Thus the cost for Phase I (SART) in BINO is $O(n^4)$. In Phase II of BINO, there are two layers of iterations outside of each non-Hanan optimization, each of which is upper-bounded by the number of buffer spaces. The combination of the total cost is $O(m^2 \cdot n^4)$. This conclusion is true for both iterative buffer insertion and iterative deletion.

The complexity of FAR-DS is same as MVERT in the outer loops. The difference is at the computation cost of reconnection part (line 6 of Fig. 9), where FAR-DS performs a search in the entire (z, ρ) space. The computation factor from searching along the ρ direction is bounded by $(\rho_{max} - 1)/\tau$, where τ is the resolution on ρ. Since the h value is swept from 1 to h_{max}, the complexity of FAR-DS is $O(h_{max} \cdot n^4 \cdot \frac{\rho_{max}}{\tau})$.

The above results only provide a loose bound, because the worst case for the quasi-binary-search along the z direction is almost impossible in practice.

Therefore, the computation cost in average case is one order lower than the above theoretic results.

5 Conclusion

A new technique for finding a minimum cost Steiner tree subject to timing specifications at the sink nodes has been presented. It has been shown that the use of non-Hanan Steiner nodes can provide noticeable benefits in improving the cost of the tree. When we extend the non-Hanan optimization to improve the performance of critical nets where both timing and wire resources are stringent, buffer insertion is shown to be a strong augmentation to the timing optimization toolkit, even with location restrictions. A combination of driver sizing and non-Hanan optimization can provide a continuous two-dimensional space. A search for the optimum in this space may be guided by properties derived from the Elmore delay model, which may have large quantitative errors but good qualitative fidelity. These properties are used to direct heuristics that use a fourth order AWE model for wire delay calculation. It is shown through experiments in [9] that combining non-Hanan optimization with buffer insertion or driver sizing can yield averagely 20% wire length reduction over SERT algorithm for timing critical nets in contrast to an average of 10% reduction from MVERT.

References

[1] J. Cong, "Challenges and opportunities for design innovations in nanometer technologies." SRC Design Sciences Concept Paper, 1997.

[2] A. B. Kahng and G. Robins, *On optimal interconnections for VLSI.* Boston, MA: Kluwer Academic Publishers, 1995.

[3] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration: the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[4] C.-K. Cheng, J. Lillis, S. Lin, and N. Chang, *Interconnect analysis and synthesis.* New York, NY: Wiley Interscience, 2000.

[5] S. S. Sapatnekar, "RC interconnect optimization under the Elmore delay model," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 392–396, 1994.

[6] J. Cong and C. K. Koh, "Interconnect layout optimization under higher-order RLC model," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 713–720, 1997.

[7] H. Hou, J. Hu, and S. S. Sapatnekar, "Non-Hanan routing," *IEEE Transactions on Computer-Aided Design*, vol. 18, pp. 436–444, Apr. 1999.

[8] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM Journal on Applied Mathematics*, vol. 14, no. 2, pp. 255–265, 1966.

[9] J. Hu and S. S. Sapatnekar, "Algorithms for non-Hanan-based optimization for VLSI interconnect under a higher order AWE model," *IEEE Transactions on Computer-Aided Design*, vol. 19, pp. 446–458, Apr. 2000.

[10] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55–63, Jan. 1948.

[11] 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*, vol. 14, pp. 1417–36, Dec. 1995.

[12] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 352–366, Apr. 1990.

[13] 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*, vol. 13, pp. 1526–1535, Dec. 1994.

[14] R. Gupta, B. Krauter, B. Tutuianu, J. Willis, and L. T. Pileggi, "The Elmore delay as a bound for RC trees with generalized input signals," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 364–369, 1995.

[15] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[16] C. L. Ratzlaff, N. Gopal, and L. T. Pillage, "RICE: Rapid interconnect circuit evaluator," in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 555–560, 1994.

[17] C. C. N. Chu and D. F. Wong, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing," in *Proceedings of the ACM International Symposium on Physical Design*, pp. 192–197, 1997.

[18] J. Cong and B. Preas, "A new algorithm for standard cell global routing," *Integration: the VLSI Journal*, vol. 14, no. 1, pp. 49–65, 1992.

[19] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI design: a systems perspective.* Reading, MA: Addison-Wesley Publishing Company, 1993.

Techniques for Timing-Driven Routing

John Lillis
Department of Computer Science
University of Illinois, Chicago, IL 60607
E-mail: jlillis@cs.uic.edu,jlillis@eecs.uic.edu

Contents

1 Introduction

With every new generation of fabrication technology for VLSI we see an increased influence of interconnect on system performance. With device scaling interconnect parasitics become increasingly influential. Issues such

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 125–153.
© 2001 *Kluwer Academic Publishers.*

as layer assignment, via resistance, wire-to-wire coupling capacitance, wire width and signal buffering now play major roles in determining signal delay.

This chapter addresses a series of design automation problems of the following general flavor: if we are to route a signal between two pins, how do we best take advantage of the various resources available (e.g., metal layers available, buffering resources, etc.) to improve signal delay? It is also the case that there tends to be a tradeoff between "cost" (e.g., routing area, buffer area, etc.) and delay. Because different nets will be competing for timing optimization resources, this notion is crucial in practice. This aspect of optimization is also emphasized throughout the chapter. The notion of cost depends on exactly how one decides to perform system-level optimization. For example, it may be wise to assign higher cost to buffers in regions where buffers are scarce, or higher cost to wiring segments in highly congested regions of the layout. Such issues are beyond the scope of this chapter, but an effort is made to maintain generality of cost models (or present means to achieve such generality).

An overview of the topics covered herein is as follows.

Basic Timing Driven Maze Routing (TDMR): This is perhaps the simplest practical formulation to examine: given technology parameters, a target routing graph in which each edge is annotated with wire resistance r and capacitance c and a source vertex s and destination vertex t; find a minimum delay path from a s to t which minimizes delay or minimizes total capacitance subject to a delay upper bound. While the first formulation may appear to be solvable by a simple shortest paths algorithm, this is not the case: because of the nonlinearity of RC delay, we need more general methods. The result is a so-called *pseudo-polynomial* algorithm which is described in detail. Since such algorithms tend to be computationally expensive, speedup techniques based on the A* algorithm are presented. This material draws from several works in the area including [20] and [12].

Incorporating Buffer Insertion: Buffer insertion is perhaps the single most powerful delay optimization tool currently available for on-chip interconnect. One approach to the buffering problem is to first determine the (rough) signal route and then insert buffers where appropriate (and possible) along this path. However, as was noted by Zhou et. al. [20], such a flow is inherently suboptimal particularly when one considers that it is often the case that buffering resources are scarce and/or not uniformly distributed. As a simple example, consider the

situation in which a design has several macro-cells. While it may be possible to route a signal over such blocks, it is impossible to buffer such signals. Thus, a router which is oblivious of this issue will likely fail to route through regions where buffering is possible. Simultaneous routing and buffer insertion is therefore proposed. Three formulations all building on the basic timing-driven maze routing algorithms are presented.

- The first objective is to simply find a buffered path which minimized delay. This can be solved through a simple modification of the TDMR algorithm.
- The second objective incorporates cost by minimizing a linear combination of delay and cost instead of simply delay. This technique supports a Lagrange Multiplier based approach to global optimization (for which the router serves as a subroutine). This formulation is also addressed through a simple modification of the basic TDMR algorithm.
- The third objective is a constrained optimization problem: minimize cost subject to a delay constraint (or alternatively, minimize delay subject to a cost bound). This objective requires a more serious overhaul of the basic TDMR algorithm and the support of additional data structures.

Dijkstra-Based Buffered Routing: Two recent papers ([15], [13]) suggest approaches to finding buffered routing paths which leverage Dijkstra's shortest paths algorithm rather then dynamic programming. These techniques are, in general, not able to easily take into consideration all of the details that the dynamic programming algorithms can (e.g., varying wire sizing resources from channel to channel) but they do tend to be very fast. As a result they may find particularly utility in early planning and/or estimation. In the discussed approaches the basic idea is to reduce the problem to that of finding paths in a *buffer graph* in which edges represent buffer-to-buffer paths. One virtue of such an approach is that it gives much more freedom in how the buffer input-to-input delays are estimated. Two such approaches will be discussed: first the problem of finding a min-delay path and second the problem of finding a tradeoff between delay and cost. The former was addressed in [15] and included novel strategies for pre-computing min-delay wire sizing configurations for buffer-to-buffer connections. The

latter is studied first as a ratio minimization problem where the objective is to minimize $\frac{D_{ref}-d(p)}{w(p)}$ where D_{ref} is a reference delay, $d(p)$ is the actual path delay and $w(p)$ is the actual path cost. Thus we have the *Delay Reduction to Cost Ratio* (DRCR) problem. This problem can also be solved quite efficiently through the use of Dijkstra's algorithm. In addition, the solutions found through this somewhat peculiar objective are further characterized with respect to the intrinsic set of non-dominated (in terms of delay and cost) paths of the problem instance. In particular, it is shown that by varying the D_{ref} parameter, any solution on the *Lower Convex Hull* of the set of non-dominated paths can be found. More generally, this notion applies to any weighted sum objective (as in the Lagrange multiplier based buffered maze routing formulation).

The chapter concludes with a discussion of the methods proposed and how they may fit into a global flow. Open questions in the area are also discussed.

2 Background and Notation

Before presenting detailed algorithms we first give necessary background on delay estimation and notational conventions.

Throughout this chapter we adopt the ubiquitous Elmore delay [6] and simple linear buffer delay estimator. In a graph model of the routing target an edge e from vertex u to v represents a candidate wiring segment. Under Elmore, the propagation delay along the wire is approximated by

$$r_e(\frac{c_e}{2} + C_v)$$

where c_e and r_e are the capacitance and resistance of the wire respectively and C_v is the total downstream capacitive load at vertex v (i.e., the total capacitance of the DC connected interconnect from v downstream).

With respect to buffer and driver delay the algorithms are also presented with a simple linear model in which the delay of a component b is estimated by

$$d_b + r_b \cdot C_L$$

where d_b is known as the *intrinsic delay*, r_b gives the buffer *output resistance* and C_L is the capacitive load on the buffer output.

A few points of discussion are in order.

- Note that the delay wire segments and buffers are context-dependent since it depends on the downstream capacitive load. This issue is essentially what makes the problem challenging.

- It is known that the Elmore delay gives an upper-bound on the actual 50% delay for RC trees [9]. To help center the error distribution, a scaling coefficient of $\ln 2$ is typically used. However, in the 2-pin problems discussed in this chapter the solutions produced are independent of any scaling factor and thus is not included in our presentation. Nevertheless, when Elmore is used for delay estimation in static timing analysis, the scaling factor is essential (as an aside, when optimizing multi-terminal nets where sinks have timing requirements, the scaling coefficient will in fact influence actual solutions [3]).

- Even with appropriate scaling coefficients, it is well known that the error of Elmore (vs. simulated delay) can be substantial in modern technologies. There are several sources of this error including Elmore's inability to consider resistive shielding and its obliviousness to such effects as signal ramp time. The linear buffer delay models exhibit similar deficiencies. Thus, it is plausible that improved solution quality may be possible by incorporating more accurate delay estimators into optimization algorithms. This issue is further complicated by the increased influence of wire-to-wire coupling capacitance in modern technologies. This general area is a topic of ongoing research. Relevant work in the area includes, [19], [18] and [11]. While such issues are beyond the scope of this chapter, the algorithms presented appear to be quite adaptable in that, provided a delay estimator can be computed incrementally from the sink to the source (perhaps with an estimate of upstream effects such as signal slew), simple generalizations can be used for the new model.

3 Basic Timing-Driven Maze Routing

The first problem we study is unbuffered maze routing. In a target routing graph we may have a variety of degrees of freedom which will produce paths of varying delay and cost (e.g., total wire area or total interconnect capacitance). These include varying interconnect parasitics from one layer to the next (e.g., a popular design style is to use upper layers for "fat", low resistance wires), the effects of via resistance and the possibility of wire sizing.

Such choices are encoded in a *multi-graph* model of the routing target – i.e., multiple edges between vertices are allowed to capture sizing options.

An important concept throught this chapter is that of *non-dominated paths.* Since a path $p : s \rightsquigarrow t$ is characterized by both its delay d and its cost c (e.g., total capacitance), there is no total order on the set of all source-to-sink paths. Consider a path $p : s \rightsquigarrow t$ with cost c and delay d. We say that p *dominates* another path p' with cost c' and delay d' if $(c < c' \text{ and } d \leq d')$ or $(c \leq c' \text{ and } d < d')$. Notationally we represent such dominance as $(c, d) \prec (c', d')$.

A path $p : s \rightsquigarrow t$ characterized as (c, d) is **non-dominated** if for all other paths $p' : s \rightsquigarrow t$ characterized by (c', d'), $(c', d') \not\prec (c, d)$. In other words, no other path is better than p in *both* dimensions. The entire set of non-dominated paths for a particular graph then becomes interesting. This notion of non-dominated paths can also be applied to sub-paths $u \rightsquigarrow t$.

For notational convenience we use the dominance relation in the context of sets of paths (or, more precisely, (c, d) pairs characterizing paths). Let P be a set of (c, d)-pairs then we have the following.

$$P \prec (c', d') \leftrightarrow \exists (c, d) \in P \text{ such that } (c, d) \prec (c', d')$$

The basic timing-driven maze routing problem can then be formalized as follows. We are given a routing target as a multi-graph $G = (V, E)$ in which each edge $e \in E$ is annotated with resistance r_e and capacitance c_e,[1] a source vertex s with output resistance r_s and intrinsic delay d_s and a destination vertex t with input capacitance c_t and an optional maximum tolerable delay d_{spec}. Our task may then be any of the following (when discussing the delay of a path the delay of the driver is always included).

(1) Find a minimum delay path $s \rightsquigarrow t$.

(2) Find a path $s \rightsquigarrow t$ which minimizes total wire capacitance subject to a delay constraint.

(3) Find the entire set of non-dominated paths (with respect to path delay and capacitance) from s to t.

[1] As an aside, c_e may be heuristically modulated to approximate the effects of coupling capacitance on a particular edge e. However, since such capacitance is not to ground, such techniques are clearly heuristic. Nevertheless, they may in practice improve solution quality.

Clearly a solution to (3) solves both (1) and (2). Further it appears that (1) and (2) are not fundamentally simpler since their currently-known solutions are based on solutions to (3). It is such an algorithm we present.

In this formulation we have a somewhat restrictive cost model since it only considers total capacitance. The computational reason for this shall become clear when examining the details of the algorithm: it turns out that capacitance will serve a dual role in incrementally computing delay and accumulating cost. On the other hand, this cost measure is quite appropriate in many situations since total capacitance is tightly bound with power consumption in CMOS technologies.

An interesting initial observation is that an algorithm which searches based only on incrementally calculated delay will not in general work correctly. The problem is that the timing-driven maze routing problem does not obey the *subpath optimality criteria* necessary for such an approach to succeed. Suppose we consider a min-delay s-to-t path p and some intermediate vertex v along the path. It is not necessarily the case that the path from v to t achieves the minimum delay among all such paths. The reason is that such a path suffix influences the delay of the path prefix up to v by way of the capacitance it presents upstream. As a result, sub-paths need to be characterized by both their delay and the capacitance they present upstream.

The algorithm TDMR for the problem is given in Figure. 1 and follows the presentation in [12]. The main ideas are as follows.

- Paths are incrementally expanded from the sink t. This expansion proceeds in lexicographic order of c and d through the maintenance of a priority queue Q. As a result of this strategy (and the monotonicity of total capacitance), suffix paths $u \rightsquigarrow t$ are examined in strictly non-decreasing order of capacitance c; by the lexicographic ordering between two paths with identical c, the one with smaller d will be examined first.

- At each vertex u in the graph we maintain a set $P(u)$, of non-dominated paths $u \rightsquigarrow t$. These paths are sorted in increasing order of c (and decreasing order of d).

- The subroutine **Candidates()** takes a non-dominated subpath $(c, d) \in P(u)$ and "augments" the solution by edges adjacent to vertex u. These new candidates are returned to the main routine which places

Subroutine: Candidates$(u, (c, d))$

```
c1   L ← ∅
c2   for each edge (u, v) incident to u
c3     if (v = s)
c4       L ← L ∪ {s, (c + c_e, d + r_e(c_e/2 + c) + r_s(c_e + c) + d_s)}
c5     else
c6       L ← L ∪ {v, (c + c_e, d + r_e(c_e/2 + c))}
c7     endif
c8   endfor
c9   return L ordered lexicographically (first by c)
```

Algorithm: TDMR

```
a1   P(t) ← {(c_t, 0)}
a2   P(u) ← ∅  ∀u ≠ t
a3   Q ← Candidates(t, (c_t, 0))
a4   while(Q ≠ ∅)
a5     (u, (c, d)) ← Dequeue(Q)
a6     if (P(u) ⊀ (c, d))
a7       P(u) ← P(u) ∪ {(c, d)}
a8       L ← Candidates(u, (c, d))
a9       for (v, (c', d')) ∈ L
a10        if (P(v) ⊀ (c', d'))
a11          Enqueue(Q, (v, (c', d')))
a12      endfor
a13    endif
a14  endwhile
a15  return P(s)
```

Figure 1: Basic Timing-Driven Maze Routing Algorithm. Output is the set of all non-dominated s-to-t paths. Path labels are propagated from the sink t toward the source s.

them in the priority queue if they are not already dominated by previous solutions (line **a10**).

- Ultimately $P(s)$ contains all of the non-dominated s-to-t paths as (c, d) pairs. Note that when a path is uncovered which reaches s, the driver delay is incorporated in line **c4**.

- Note that since members of $P(u)$ are uncovered in increasing order of c and decreasing order of d, the algorithm can be modified to stop early once a delay spec is met or a capacitance upper bound is exceeded.

- Because of the monotonicity of c, $P(s)$ is built up starting from the lowest-c solution and progressing to the min-d solution. A restructuring of the algorithm would expand solutions first by d and secondarily by c. Because of the monotonicity of d, the correctness will still hold, but the min-delay solution will be uncovered first.

Using a standard binary heap for a priority queue, almost the entire pseudo-code can be easily translated to an implementation. However, the testing of dominance in lines **a6** and **a10** of the algorithm warrants some discussion. For notational purposes, let us focus on the test in line **a6** where we wish to determine if $P(u) \not\prec (c, d)$. Let $P(u) =< (c_1, d_1), ..., (c_k, d_k) >$ arranged in increasing order of c_i. By virtue of the fact that paths are uncovered in non-decreasing order of capacitance, we know that $c_i \leq c$. Thus the test $P(u) \not\prec (c, d)$ is equivalent to testing if $\forall (c_i, d_i) \in P(u), d_i < d$. Since d_k is the smallest among the d_i's, this test can be done in constant time. In practice it is a simple matter of comparing with the tail element of a linked list.

The correctness of the algorithm given by the following.

Theorem 3.1 *At the termination of algorithm TDMR, $(c, d) \in P(u)$ if and only if there exists a non-dominated path $u \rightsquigarrow t$ with capacitance c and delay d.*

Proof. Let $P'(u)$ be the (true) set of non-dominated paths $u \rightsquigarrow t$. We first show that $(c, d) \in P'(u) \Rightarrow (c, d) \in P(u)$. Suppose this is not the case. Let (c, d) be a lexicographically minimal element over all u where $(c, d) \in P'(u)$ but $(c, d) \notin P(u)$. First we observe that (c, d) cannot be $(c_t, 0) \in P'(t)$ since it is added to $P(t)$ as a basis step. Thus, the path must contain at least one edge; let v be the next vertex after u on the path. Let this subpath $v \rightsquigarrow t$ be (c', d'). Since $(c', d') \in P'(v)$ is lexicographically less than (c, d)

we know that $(c', d') \in P(u)$. This implies that at some point $(v, (c', d'))$ was dequeued at line **a5** and that after expansion $(u, (c, d))$ was enqueued at line **a11** (since it is a non-dominated solution). Further, $(u, (c, d))$ will eventually be dequeued and placed into $P(u)$ yielding a contradiction.

We now know that $P'(u) \subseteq P(u)$ and must show that $P(u) \subseteq P'(u)$. Since $P'(u)$ is the set of *all* non-dominated paths $(u, (c, d))$, it is sufficient to show that $P(u)$ is *mimimal* – i.e., that for any $(c, d), (c', d') \in P(u)$, $(c, d) \not\prec (c', d')$ and $(c', d') \not\prec (c, d)$. This fact follows from the lexicographically non-decreasing order in which elements are added to $P(u)$ and the dominance test of line **a6**. □

The algorithm is pseudo-polynomial in that the run time depends on the numerical parameters of the problem (in particular edge capacitance) as well as the problem size ($|V|$ and $|E|$). We give a run-time analysis based on the following assumptions.

- The capacitive values c_e are given as integers. This presumes that some degree of discretization may be performed. Note however, that the correctness of the algorithm remains if no such assumption is made.
- The node degree in the target routing graph is bounded by a constant. This assumption matches VLSI applications in which there are typically two edges on the same layer and candidate vias to adjacent layers. In the event wire-sizing is possible, we presume there are only a handful of choices which appears to be the prevailing assumption in practice.

To capture the effect of the capacitive values on run-time, let $U = \sum_{e \in E} c_e$. This allows the following observations.

- $|P(u)| \leq U$. The bound follows since U gives an upper bound on the number of distinct c's and there are no duplicate c's in $P(u)$.
- $|Q| = O(U \cdot |V|)$. This results from the constant node degree. When a solution is dequeued, it can only introduce a constant number of additional solutions into the queue. Since only non-dominated solutions induce additional solutions a vertex u can induce at most $O(|P(u)|)$ queue entries. Combining with the previous observation gives the bound.

The overall runtime is then given by the following theorem.

Theorem 3.2 *The running time of algorithm TDMR is* $O(U \cdot |V| \log(U \cdot |V|)$.

Proof. The total number of enqueue/dequeue operations is $O(U \cdot |V|)$; further there are $O(U \cdot |V|)$ dominance tests each of which takes $O(1)$ time. Using a simple binary heap data structure, gives an $O(\log(U \cdot |V|))$ bound for individual enqueue/dequeue operations. This gives the overall bound of $O(U \cdot |V| \log(U \cdot |V|)$. □

As is often the case with pseudo-polynomial bounds, this expression may be quite pessimistic in practice. For instance it is extremely unlikely that $|P(u)|$ will approach $|U|$ in practice.

Speedup Techniques

A speedup enhancement of the basic TDMR algorithm was given in [12]. The technique can be seen as an application of the A* method [10]. A* based approaches attempt to make path searching more goal-directed by biasing the expansion of vertices toward those which are, by some measure, in the direction of the destination vertex (in our case, the source vertex). The measure which is used to bias the search is typically called a *heuristic cost.* The heuristic cost usually gives a lower bound on the length of the "tail" of a path (where we already have a prefix). Instead of expanding paths in order of the prefix cost, A* expands paths in order of the prefix cost plus the lower-bound cost. Thus path prefixes which are estimated to lead to short overall paths are expanded first and the expansion of less promising paths is suppressed.

To get an idea of the potential of an A* type approach in TDMR, consider the case in which we have significant freedom in wire sizing. It is known that ideally wire widths grow narrower toward the sink since fat wires near the sink produce excessive load on the long upstream interconnect. However, a candidate subsolution $(c, d) \in P(u)$ for some u near the sink t may indeed use wide wires and still be non-dominated. This is a result of the algorithm being local in nature and oblivious of the actual source s. A* attempts to correct this situation.

To adapt A* to TDMR we introduce the following generic lower-bounding functions.

> $c_{\min}(v)$: a lower-bound on the total capacitance of any subpath from s to v.

$d_{\min}(v, c)$: a lower-bound on the path delay of any path s to v where we have a downstream load of c at v. This includes driver delay.

We then store candidate solutions in the priority queue as $(u, (c+c_{\min}(v), d+d_{\min}(v, c)))$ rather than $(u, (c, d))$ in the generic algorithm.

Computation of $c_{\min}(v)$ can be done by a simple pre-process: run Dijkstra's shortest paths algorithm starting at vertex s with edge weights c_e.

Computation of $d_{\min}(v, c)$ is somewhat more involved. In [12], the following strategy was suggested. Let l_e be the actual distance between the endpoints of edge e. We define $l_{\min}(v)$ as

$$l_{\min}(v) = \min_{\text{paths } p:s \rightsquigarrow v} (\sum_{e \in p} l_e)$$

which can also be pre-computed with another pass of Dijkstra's algorithm. There has been some recent work (e.g., [8], [2]) which can be leveraged in getting an effective bound $d_{\min}(v, c)$. Those papers studied optimal tapering functions (wire-width as a function of distance from the driver). It was suggested that a simple practical technique for $d_{\min}(v, c)$ was to pre-compute a technology specific table which, given wire-length, driver resistance and downstream capacitance would give the minimum possible delay when arbitrary tapering was allowed. This then clearly becomes a lower-bound on the delay we can achieve in the routing target. Further, it is likely to be reasonably tight.

The overall speedup achieved by this technique as reported in [12] were impressive. Speedups of up to 300× were achieved versus the generic TDMR algorithm.

4 Incorporating Buffer Insertion

While such techniques as wire sizing and layer assignment implicitly utilized by the TDMR algorithm of the preceding section, the techniques are limited in the amount of delay reduction possible and may not be the most cost-effective means of delay reduction. An interesting reference in this area is [1] in which Alpert experimentally studied the issue.

On the other hand, *buffer insertion* does generally produce very cost-effective delay reduction. The effectiveness of buffer insertion is due to the capacitive decoupling effect of buffers. The result is that the delay of long

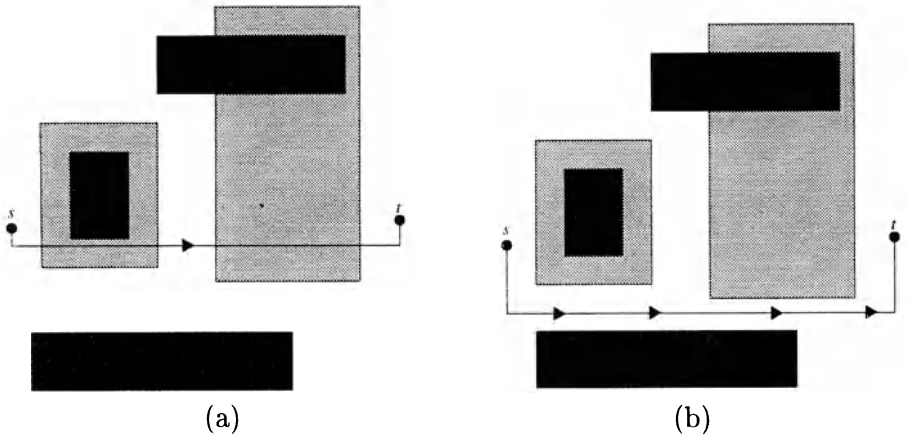


Figure 2: Buffered routing in the context of a floorplan. Dark boxes completely disallow buffering *and* routing; shaded regions may be routed through, but buffering is disallowed. Scenario (a) shows a post-route buffer insertion strategy while (b) is a simultaneous approach. A small increase in wire-length enables much more flexibility in buffering options.

lines is "linearized" by segmentation.[2]

Thus, it can reasonably be argued that it makes sense to integrate buffer insertion with timing driven routing. While it is possible to perform buffer insertion as a two-phase process in which a route is first determined and then buffers are inserted along the path, this approach is clearly suboptimal as pointed out by Zhou et. al. in [20]. This suboptimality results from the fact that buffers in general cannot simply be inserted *anywhere* and thus the signal path affects where along the path buffers can be inserted. As an example from [20], consider the situation in which we have pre-placed macro-cells. It may be possible to route over such cells, but unlikely that we can insert buffers in such regions. As a result, we may prefer to take a route which does not cross the macro-cell or even detours a bit to enable the insertion of a buffer. This situation is illustrated in Figure 2.

This section examines three formulations of the buffered routing problem. The first two formulations can be solved by a straightforward generalization of the basic TDMR algorithm. The simplest formulation is simply

[2]Buffers are also essential in decoupling off-path capacitance in multi-pin nets, but this topic is beyond the scope of this chapter.

targeted at finding the minimum delay path irrespective of cost (number of buffers, wiring capacitance, etc.). The second formulation also produces just a single path, but the path minimizes a composite objective including both delay and cost with appropriate scaling coefficients. The third formulation is a constrained delay minimization problem which requires more serious re-tooling of the basic algorithm.

Suppose we are given a buffer library B where each buffer $b \in B$ has the following properties.

c_b: input capacitance.

r_b: output resistance.

d_b: intrinsic delay.

w_b: nominal cost.

In many full-chip optimization flows, the actual location at which a buffer is inserted may influence the cost. For instance inserting a buffer in a region where buffering resources (white space) are scarce might be considered to cost more than inserting the same buffer in another region with more white space. Further, it often makes sense to have flexibility in how costs are assigned to routing edges for similar reasons (e.g., routing congestion).[3] To address this situation we introduce the following annotations of the target routing graph.

w_e: cost of routing edge $e \in E$.

w_v: scaling coefficient applied to the insertion of a buffer at vertex $v \in V$. The cost of placing buffer $b \in B$ at vertex v is then said to be $w_v \cdot w_b$.

We also note the generality of this formulation in that in the event that inserting a buffer at some vertex v is simply not possible, we can effectively set $w_v = \infty$ (in practice, this can be handled with a flag and buffered options simply not considered.)

4.1 Weighted Sum Minimization

Given the parameterization of the preceding section, we can then say that, for the second problem formulation, the objective is to minimize a weighted

[3]This is a weakness of the basic TDMR algorithm in that it only considers cost to be total capacitance.

sum of cost and delay: $A_1(w(p))+A_2(d(p))$. A solution to this problem gives us a solution to the problem of finding a min-delay buffered path simply by setting A_1 to 0.

Figure 3 gives pseudo-code solving the weighted-sum buffered maze routing problem. This presentation builds on that of [3] and benefited from input from J. Fishburn [7]. A few points deserve discussion.

- Instead of using (c, d) pairs as the "signature" for a candidate subpath, we use (w, c) pairs where w is a cumulative weighted sum of cost and delay and c is the "visible" capacitance – i.e., the capacitance up to the next device input.

- Any cost-delay tradeoffs among paths must be captured in the w-value. In contrast, the basic TDMR algorithm used c for dual purposes: delay calculation and accumulation of cost. In this case, it is simply carried along for dominance testing and delay calculation.

- Capacitance c is no longer a monotone function as we travel from sink to source. This is a direct result of the decoupling effect of buffer insertion – once a buffer is inserted we can no longer "see" its downstream load. This has a serious implication: if solutions are expanded in order of c, they are not necessarily expanded in non-decreasing order as in the basic algorithm. For this reason, we expand solutions in order of w which is monotonically non-decreasing as we travel from the sink to the source. This preserves the monotone wavefront property so important in many path searching algorithms.

- Since solutions are uncovered in non-decreasing order of w, the *first* dequeued solution which includes the source s is the optimal path and thus the algorithm can terminate as in line **a6**. Further, the dominance tests remain a simple matter of comparing with the tail entry of a linked list.

The correctness of the algorithm follows from a similar argument as given in Theorem 3.1. However, strictly speaking, we cannot guarantee that the path discovered is *simple*. This was an observation made by Fishburn [7] and is discussed in the next subsection.

The run time is slightly more difficult to pin down. The size of the candidate sets $P(u)$ is limited by the distinct number of weighted sum values of interest. Supposing that these weighted sum values are integers, let w_{opt} be the cost of the solution returned by the algorithm. We can then

```
Subroutine: Candidates(u, (w, c))
c1   L ← ∅
c2   for each edge (u, v) incident to u
c3     d' ← r_e(c_e/2 + c))}
c4     w' ← w + A_1 · w_e + A_2 · d'
c5     if (v = s)
c6       L ← L ∪ {s, (c + c_e, w' + A_2(r_s(c_e + c)d_s)}
c7     else
c8       L ← L ∪ {v, (c + c_e, w + w'))}
c9       for b ∈ B
c10        L ← L ∪ {v, (c_b, w' + A_1 w_b w_v + A_2(d_b + r_b c)))}
c11    endif
c12  endfor
c13  return L ordered lexicographically (first by w - not by c)
```

```
Algorithm: TDMR-Buff-Weight
a1   P(t) ← {(c_t, 0)}
a2   P(u) ← ∅  ∀u ≠ t
a3   Q ← Candidates(t, (c_t, 0))    // Q ordered first by w
a4   while(Q ≠ ∅)
a5       (u, (w, c)) ← Dequeue(Q)
a6       if u = s return w
a7       if (P(u) ⊀ (w, c))
a8         P(u) ← P(u) ∪ {(w, c)}
a9         L ← Candidates(u, (w, c))
a10        for (v, (w', c')) ∈ L
a11          if (P(v) ⊀ (w', c'))
a12            Enqueue(Q, (v, (w', c'))
a13        endfor
a14      endif
a15  endwhile
```

Figure 3: Buffered Timing-Driven Maze Routing Algorithm for a weighted-sum formulation. At termination, the first element added to $P(s)$ encodes the optimal path. Path labels are propagated from the sink t toward the source s. Note that the priority queue Q is ordered by w first, not c.

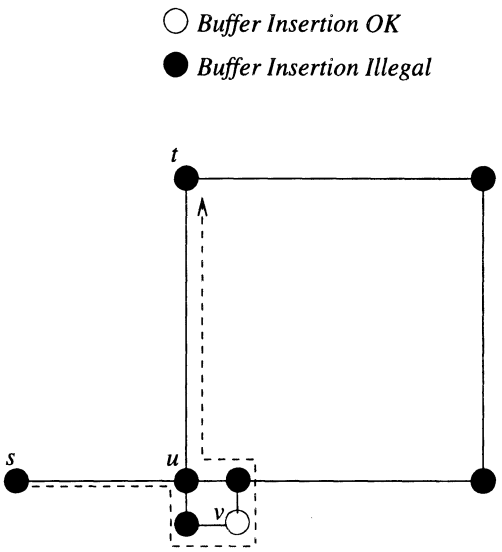


Figure 4: Anomalous behavior of buffer insertion. Nothing prevents the path from intersecting itself.

upper bound $|P(u)|$ by w_{opt} since the weighted sums are non-negative and all other entries in sets $P(u)$ are no larger than w_{opt}. By noticing that the number of enqueue operations is $O(|B|)$ under the constant degree assumption, we can then conclude that $Q = O(|B||V|w_{opt})$. Thus, the enqueue operations will dominate the overall run time and we have an overall bound of $O(|B||V|w_{opt} \log((|B||V|w_{opt})))$.

Once again however, this run-time should be taken with a grain of salt since it is a rather crude upper-bound. We note also that in practice, it is even likely that the buffered algorithm will be faster than the unbuffered algorithm since the ability to insert buffers will enable the pruning of many high-capacitance suffix paths which are not in general eliminated without buffer insertion and are not considered in the run time analysis.

Discussion

An interesting property of this formulation as observed in [3] and independently by Fishburn [7] is that a source-to-sink (top-down) algorithm is equally viable. The sole role of c in the presented algorithm is to enable cor-

rect delay calculation. In a top-down approach we would accumulate path resistance (from the driving buffer) to serve this role. Note that, as in the case of capacitance, resistance is not monotone since inserting a buffer "cuts off" the preceding resistance. It is not currently clear whether a top-down approach would behave significantly different in terms of run-time compared with the bottom-up approach. However, note that the availability of both top-down and bottom-up methods makes possible a bi-directional search mechanism in which propagation of paths from the source is interleaved with propagation from the sink. Such a scheme seems likely to produce significant speedup (particularly when one considers that the number of labels at a vertex tends to grow with the distance of that vertex from the propagation source) and would make an for an interesting implementation study.

Perhaps a more significant curiosity was also observed by Fishburn [7]. Since capacitance in the presented algorithm (and resistance in the top-down approach) is non-monotone (due to the decoupling effect of buffers), it is possible that an anomily may occur in which a path inappropriately re-uses a vertex – i.e., that the paths are not *simple*. This phenomenon is illustrated in Figure 4. We leave it as an open question whether this problem can be addressed efficiently in a theoretically robust way. It is difficult to say how serious this problem is in practice. If on the one hand, the algorithm is being used for timing-driven global routing of timing critical nets (perhaps with layer assignment) it is likely that vertices in the graph correspond to global routing tiles rather than electrically connected nodes and thus a detailed router will resolve the issue. If on the other hand detailed routing and buffer insertion are being performed simultaneously, the problem may be of more concern.

4.2 Constrained Optimization with Buffers

In the constrained optimization formulation of the problem, we have a given parameter d_{spec} and our objective is to minimize solution cost (weight) subject to total path delay being no greater than d_{spec}.

A solution to this problem follows the same basic framework as the preceding algorithms with the main key difference being that we carry three parameters instead of two. A path suffix will be characterized by the triple (w, c, d) where w is the accumulated cost, c is the visible capacitance and d is the downstream delay.

The framework is essentially the same as before. We expand solutions

in lexicographically non-decreasing order: first ordered by w, secondarily by d, thirdly by c. In sets $P(u)$ we store non-dominated solutions where dominance is determined by all three dimensions: a path (w, d, c) from u to t is non-dominated if no other path is better in all three dimensions. Pseudo-code appears in Figure 5. A key difference however is that the third parameter makes testing of dominance in lines **a6** and **a10** non-trivial.

To support these tests we need to efficiently determine, given a non-dominated set P and a candidate solution (w, d, c) whether $P \not\prec (w, d, c)$:

$$P \not\prec (w, d, c) \Leftrightarrow \forall (w', d', c') \in P,\ w < w' \text{ OR } d < d' \text{ OR } c < c'.$$

Equivalently an efficient test for $P \prec (w, d, c)$ will do the job:

$$P \prec (w, d, c) \Leftrightarrow \exists (w', d', c') \in P,\ w \geq w' \text{ AND } d \geq d' \text{ AND } c \geq c'.$$

By virtue of the fact that candidate solutions are expanded in non-decreasing order of w, we observe that the first inequality ($w \geq w'$) holds for *every* member of P. Thus the problem reduces to determining if *some* member of P is at least as good as (w, d, c) in the d and c dimensions. The existence of such a member of P can of course be determined through a linear scan of P. However, with use of an appropriate data structure the problem can be solved in $O(\log(|P|)$ time.

Such a data structure was given in [17] (for the optimization of static routing topologies) and is adapted here. We store the members of P in a binary search tree ordered by d (w is ignored due to the reasoning above). At each node t of the tree we store the following.

t.d: The associated d-value.

t.c: The associated c-value.

t.c_l_min: The smallest c-value in the left subtree. If a vertex has no left child, this value is considered to be ∞.

We then want to support queries in which we are given a candidate (d, c) if there is some entry (d', c') already in the tree where $c' \leq c$ and $d' \leq d$; if so, then the candidate is sub-optimal and if not, the solution is non-dominated. The recursive rules for traversing the tree are as follows (in the base case where the tree is NULL, the solution is clearly non-dominated).

Subroutine: Candidates$(u, (w, d, c))$

```
c1   L ← ∅
c2   for each edge (u, v) incident to u
c3     d' ← d + r_e(c_e/2 + c))}
c4     w' ← w + w_e
c5     if (v = s)
c6       L ← L ∪ {s, (w', d' + d_s + r_s(c + c_e), c + c_e)}
c7     else
c8       L ← L ∪ {v, (w + w', d', c + c_e))}
c9       for b ∈ B
c10        L ← L ∪ {v, (w' + w_b w_v, d' + d_b + r_b(c + c_e), c_b))}
c11    endif
c12  endfor
c13  return L ordered lexicographically
```

Algorithm: TDMR-Buff-Const

```
a1   P(t) ← {(0, 0, c_t)}
a2   P(u) ← ∅  ∀u ≠ t
a3   Q ← Candidates(t, (0, 0, c_t))    // Q ordered first by w
a4   while(Q ≠ ∅)
a5       (u, (w, d, c)) ← Dequeue(Q)
a6       if (P(u) ⊀ (w, d, c))
a7         P(u) ← P(u) ∪ {(w, d, c)}
a8         L ← Candidates(u, (w, d, c))
a9         for (v, (w', d', c')) ∈ L
a10          if (P(v) ⊀ (w', d', c'))
a11            Enqueue(Q, (v, (w', d', c'))
a12        endfor
a13      endif
a14  endwhile
```

Figure 5: Buffered Timing-Driven Maze Routing Algorithm for a constrained optimization formulation. At termination, only solutions in $P(s)$ which are non-dominated with respect to w and d are of interest since c is merely an artifact of the algorithm.

case	condition	action
(1)	$d < t.d$ AND $c \geq t.c\_l\_min$	$t \leftarrow t$.left
(2)	$d < t.d$ AND $c < t.c\_l\_min$	**return** "Non-Dominated"
(3)	$d \geq t.d$ AND $c \geq \min(t.c\_l\_min)$	**return** "Dominated"
(4)	$d \geq t.d$ AND $c < \min(t.c, t.c\_l\_min)$	$t \leftarrow t$.right

In cases 1 and 2, any solution dominating (d, c) must be in the left subtree: in case 1 we can't be sure if such a solution exists; in case 2, we know that c is smaller than all c's in the left subtree and thus, (d, c) must be non-dominated. In case 3 we know that either the root or *some* entry in the left subtree dominates (c, d). In case 4 the root and left subtree cannot eliminate the candidate, so we explore the right subtree.

An example of this data-structure appears in Figure. 6. Keep in mind that underlying the (d, c) values in the tree are w values and thus the (d, c) values are not in general minimal (non-dominated with respect to themselves), but the corresponding (w, d, c) triples are. Consider a query in which we wish to determine if $(26, 9)$ is dominated. We will apply case 3 at the root since some solution in the left subtree dominates. Suppose instead the query is $(26, 8)$; in this case we traverse the tree all the way to vertex $(27, 12)$ and finally a NULL subtree to conclude that it is non-dominated. If the query is $(24, 6)$, we apply case 2 at the root and declare the solution non-dominated.

The idea then is that instead of maintaining a simple linked list to store $P(u)$, we instead use the augmented binary tree. The strategy can be implemented such that queries and insertions are always logarithmic in the size of the tree by applying a tree balancing scheme such as Red-Black trees [5] (updating of the augmenting values during the rotation operations is straightforward and local).

With respect to running time we once again have a pseudo-polynomial bound. First we would like to bound the size of the $P(u)$ sets. Let $W_{\max}$ be the largest total cost seen during the algorithm and $C_{\max}$ be an upper bound on any individual load capacitance (both being integers). We can conclude that $|P(u)| = O(W_{\max} C_{\max})$ since for any distinct pair of w and c values, at most one d value is of interest. Again using the constant degree assumption, we then conclude that $|Q| = O(|V||B|W_{\max} C_{\max})$. Since this dominates the size of the individual sets $P(u)$, the time for enqueues and dequeues will dominate operations of the augmented binary trees giving an overall run time of $O(|V||B|W_{\max} C_{\max} \log(|V||B|W_{\max} C_{\max}))$.

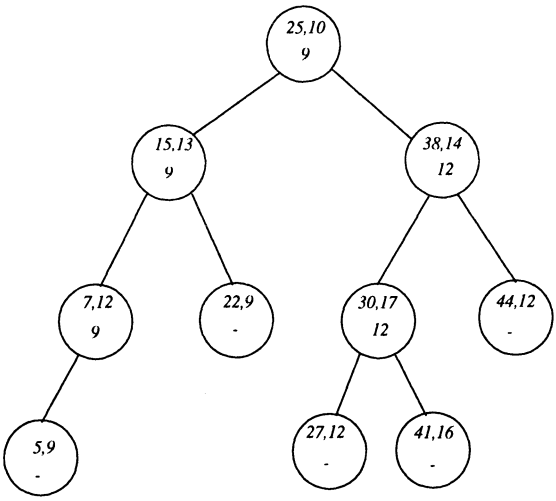


Figure 6: Augmented binary search tree for detecting dominated/non-dominated candidate solutions. Vertex labels are d, c pairs on top and the lower number is the min c-value in the left subtree.

4.3 Discussion

In this section we have seen two approaches to buffered maze routing. The first adopted the objective of minimizing a composite objective function and thus produced a single path. This approach subsumed the case where we simply want the min-delay configuration. The second approach solved a constrained optimization problem or more generally produced an entire family of non-dominated paths. This tradeoff curve provides "sensitivity" information which may be useful in a global optimization mechanism. The price for obtaining this additional information was higher computational implementation complexity. The preferred approach may very well depend on higher-level issues such as the strategy being adopted for full-chip optimization. The next section presents a result which gives a theoretical relationship between such tradeoff curves and solutions found under a composite weighted sum objective. In particular, we will see that any solution on the *lower convex-hull* of the set of all non-dominated paths can be identified by re-weighting and solving the composite objective.

Either algorithm can be modified to support the use of inverters as buffers. We need to ensure that any s-to-t path must have an even num-

ber of inverters. The algorithm modification follows [17]. We maintain at each vertex v two sets $P^+(v)$ which contains paths with an even number of inverters and $P^-(v)$ which contains paths with an odd number of inverters.

Another important issue is speedup techniques. In the unbuffered maze routing problem it was suggested that by applying lower-bounds on path delay and cost, one could use the the A* approach to accelerate the algorithm significantly. The same concept can certainly be applied in the case of buffered maze routing. Recent work in closed forms for minimum delay uniform buffered lines (e.g., [4] can be used toward this end).

5 Dijkstra-Based Buffered Routing

Recently there have been two papers ([15], [13]) which propose techniques for buffered routing which do not rely on Dynamic Programming as the preceding presentations. In each of these papers a *buffer-planning graph* or *buffer-graph* is constructed in which vertices represent candidate buffers locations and the source and sink and edges represent buffer-to-buffer connections. The ideas are summarized as follows.

- We are given a target routing graph with a vertex set $V = V_a \cup V_b$ where buffer insertion is permitted at vertices $v \in V_b$, but forbidden at vertices $u \in V_a$.
- We are also given a buffer library B.
- We compute shortest path lengths $l(u, v)$ for each pair $u, v \in V_b \cup \{s, t\}$.
- We then utilize a delay estimator $d(l, b_1, b_2)$ which estimates the delay from the input of a buffer $b_1 \in B$, through an interconnect of length l to the input of a buffer $b_2 \in B$.
- For each pair $b \in B$ and $v \in V_b$, we introduce a vertex in the buffer graph. To these we add the source vertices s and t.
- The buffer graph is in general dense with directed edges between all pairs of vertices (in practice some of these edges can be filtered out). Suppose we have an edge from vertex u to v with buffers b_1 and b_2 at u and v respectively. We label this edge with $d(l(u, v), b_1, b_2)$.

This construction is illustrated in Figure 7. As a result of the construction, delay is *context independent* since delays are always from buffer input

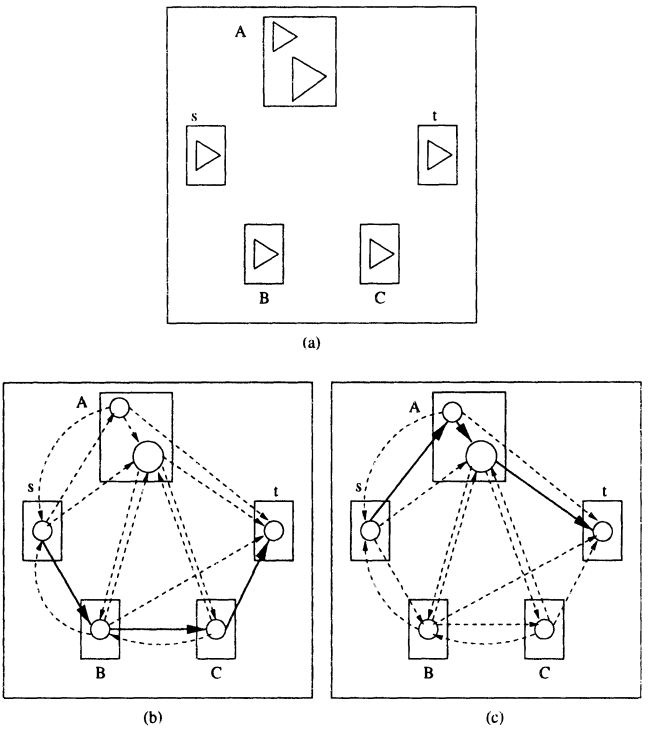


Figure 7: Example buffer graph. In (a) we have the available buffer stations; (b) and (c) show two candidate paths in the corresponding buffer graph. Note that some edges may be eliminated in practice (e.g., (s, t) if the distance exceeds a technology specific threshold. Buffer block A enables local cascading.

to input. It has the additional virtue that the delay function need not be based on Elmore delay; any delay estimator may be used. Aiming toward minimizing path delay Lai and Wong [15] propose a refinement of this approach in which the delay estimator uses a lookup table to give the minimal achievable delay if the buffer-to-buffer wire were optimally tapered. An additional feature of this construction is that it implicitly allows cascading of buffers from small buffers to larger at an insertion point.

The net effect of this construction is that finding a minimum delay from s to t now becomes simply that of finding a minimum weight path in the buffer graph. The buffers inserted on the path are implicitly determined

by the vertices on the path. The details of the buffer-to-buffer routes are then deferred to another routing process or the paths from the all-pairs shortest-paths computation used to construct the buffer graph.

In [13] essentially this same buffer graph construction was applied. However it was argued that the minimum delay path is rarely the solution which makes the most engineering sense. In general, there is a tradeoff between delay and cost (e.g., total capacitance, area, etc.). It is often the case that if we can tolerate a small increasing in delay, we can save a great deal in cost versus the min delay path. This has significant practical implications since many signal nets may be competing for the same scarce buffering resources.

To capture this tradeoff between delay and cost without resorting to dynamic programming, [13] proposed the *Delay Reduction to Cost Ratio* (DRCR) formulation. In this formulation each edge in the buffer graph is additionally annotated with a cost w which accounts for the interconnect cost and the cost of the destination buffer. This cost label may be determined by whatever means desired; for instance if buffering resources in the region are scarce, the cost may be increased. The DRCR then takes a reference delay D_{ref}, and the objective is to find a path p which maximizes

$$\frac{D_{ref} - d(p)}{w(p)}$$

where $d(p)$ is the path delay and $w(p)$ is the path cost. The numerator encourages delay reduction and the numerator encourages lower cost.

This formulation can be solved by a labeling scheme similar to algorithms for the minimum time-to-profit problem [16]. Suppose we have a conjecture I of the optimal ratio. We then label each edge in the graph with the weight $Iw(e)+d(e)$ and find a min-weight path from s to t in the re-weighted graph. The results of this shortest path computation is interpreted as follows.

- If the weight of the path $w(p)$ is equal to D_{ref}, we have found the optimal path and I is the minimum achievable ratio.
- If on the other hand, $w(p) < D_{ref}$, our conjecture is too small (this path actually achieves a better ratio) and I must be increased.
- If $w(p) > D_{ref}$, our conjecture is too large (no path can achieve this ratio), and I must be decreased.

The strategy then is to perform binary search on I, performing a shortest path computation for each value. If D is the delay of the min cost path,

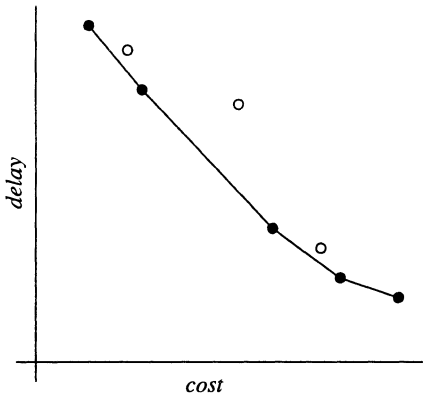


Figure 8: Lower Convex Hull. Figure shows set of all non-dominated paths. Solid points are on the LCH; open points are not.

then the number of iterations is bounded by $O(\log D)$ giving a strongly polynomial run time bound. Thus, for a given D_{ref} we can efficiently solve the DRCR problem.

In [13], an effort was then made to characterize the solutions to the DRCR problem. The given buffer graph has an intrinsic set of non-dominated paths which can be revealed through a pseudo-polynomial algorithm for the *shortest weight-constrained path* problem (such an algorithm is quite similar to the TDMR algorithm; see [14]). Such a set of solutions appears in Figure 8. The *Lower Convex Hull* (LCH) of this set is the set of solid points. The following theorem from [13] reveals the relation between solutions found by the DRCR algorithm and LCH of the non-dominated paths.

Theorem 5.1 *A path (w, d) is on the LCH of non-dominated paths if and only if $\exists D_{ref}$ for which (w, d) maximizes the DRCR.*

This would appear to be a nice property since the solutions which make the most cost-effective use of the available resources versus solutions off the LCH. Further, it was noted in [13] that, a coefficient I maximizes DRCR for *some* D_{ref} and thus, the LCH can be probed directly by varying I.

Further, the theorem shows the same relationship between the two algorithms for buffered maze routing presented in section 4. In particular, by varying the coefficients in the weighted sum formulation (which used pairs),

we can probe the LCH of the tradeoff curve produced by the constrained optimization algorithm (which used triples).

6 Conclusion

This chapter has focused on the joining of routing and delay optimization for two pin nets. A variety of formulations were studied: basic timing-driven maze routing, variants with buffer insertion and a class of faster buffering algorithms appropriate for earlier design stages.

The techniques surveyed would seem to provide an efficient and flexible set of tools for full-chip optimization. Nevertheless, more research is required to fully understand timing closure and resource allocation issues there. The techniques may also be effective tools in designing effective timing driven heuristics for routing multi-pin nets.

References

[1] C. J. Alpert, A. Devgan, J. P. Fishburn, S. T. Quay, Interconnect Synthesis Without Wire Tapering, *IEEE Transactions on Computer-Aided Design,* Vol. 20, No. 1, Jan. 2001, pp. 90-104,

[2] C.-P. Chen, Y.-P. Chen, D.F. Wong, Optimal Wire-Sizing Formula under the Elmore Delay Model, *Proc. Design Automation Conference*, pp. 487-490, June 1996.

[3] C.-K. Cheng, J. Lillis, S. Lin and N. Chang, Interconnect Analysis and Synthesis, (Wiley Interscience), 2000.

[4] C. C. N. Chu and D. F. Wong. Closed Form Solution to Simultaneous Buffer Insertion/Sizing and Wire Sizing. *Proc. Intl. Symp. on Physical Design*, pp. 192-197, 1997.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, Introduction to Algorithms, (McGraw Hill), 1989.

[6] W.C. Elmore, The transient response of damped linear networks with particular regard to wide-band amplifiers, *Journal of Applied Physics*, 19(1): pp. 55-63, Jan. 1948.

[7] J.P. Fishburn, private communication (2000).

[8] J.P. Fishburn, C.A. Schevon, Shaping a distributed-RC line to minimize Elmore delay, *IEEE Trans. Circuits Syst. I,* vol. 42, pp. 1020-1022.

[9] R. Gupta, B. Tutuianu and L. Pileggi, The Elmore Delay as a Bound for RC Trees with Generalized Input Signals, *IEEE Transactions on Computer-Aided Design*, Vol. 16, No. 1, pp. 95-104, January 1997.

[10] P.E. Hart, N.J. Nilsson, B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. Syst. Cybern.*, vol. SCC-4, pp. 100-107, July 1968.

[11] J. Hu and S. S. Sapatnekar, Algorithms for Non-Hanan-based Optimization for VLSI Interconnect under a Higher Order AWE Model, *IEEE Transactions on CAD*, Vol. 19, No. 4, pp. 446-458, April 2000.

[12] S.-W. Hur, A. Jagannathan, J. Lillis, Timing-Driven Maze Routing, *IEEE Trans. on CAD*, vol. 19, no. 2, pp. 234-241.

[13] A. Jagannathan, S.-W. Hur, J. Lillis, A Fast Algorithm for Context-Aware Buffer Insertion, *Proc. Design Automation Conference*, pp. 368-373, June 2000.

[14] H.C. Joksch, *The Shortest Route Problem with Constraints J. Math Anal. Appl.*, 14, pp. 191-197, 1966.

[15] M. Lai, D.F. Wong, Maze Routing with Buffer Insertion and Wiresizing, *Proc. Design Automation Conference*, pp. 374-378, June 2000.

[16] E. Lawler, Combinatorial Optimization, Networks and Matroids, (Hold, Rinehart, Winston), pp. 94-97, 1976.

[17] J. Lillis, C.-K. Cheng, T.-T. Y. Lin, Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model, *IEEE J. Solid-State Circuits*, vol. 31, no. 3, pp. 437-447, 1996.

[18] J. Qian, S. Pullela and L.T. Pillage, Modeling the "Effective Capacitance" of RC Interconnect, *IEEE Transactions on Computer-Aided Design*, pp. 1526-1535, December 1994.

[19] J. Lillis, P. Buch, Table-Lookup Methods for Improved Performance Driven Routing, *Proc. Design Automation Conference*, pp. 368-377, June 1998.

[20] H. Zhou, D.F. Wong, I.-M. Liu, A. Aziz, Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations, *Proc. Design Automation Conference*, pp. 96-99, June 1999.

Interconnect Modeling and Design With Consideration of Inductance

Lei He
Department of Electric and Computer Engineering
University of Wisconsin, Madison, WI 53706
E-mail: lhe@ece.wisc.edu

Contents

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 155–190.
© 2001 *Kluwer Academic Publishers.*

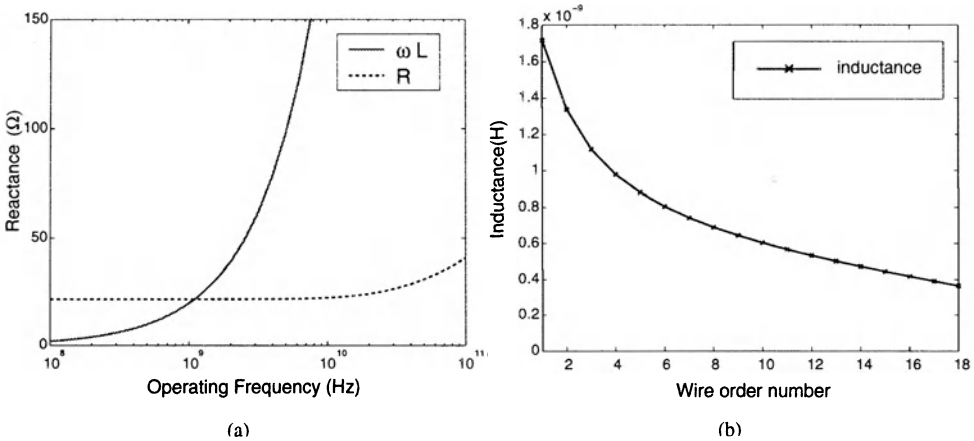


Figure 1: (a)Characteristics of resistance and reactance due to inductance; (b) Characteristics of self inductance and mutual inductance.

1 Introduction

Given the growing importance of interconnects in performance, reliability, cost, and power dissipation for high-performance circuits and systems, interconnect modeling and optimization has been an active research area [1]. However, most existing work on interconnect modeling and optimization assumes an RC interconnect model, which becomes increasingly inadequate as the on-chip inductive effect gains prominence in gigahertz designs. A simple rule of thumb is that the inductance should be considered if resistance R and reactance ωL have similar values, where L is inductance and $\omega = 2\pi f$ with f being the operating frequency. In Figure 1(a), we compare R and ωL under different operating frequencies. We used the three-dimensional electromagnetic field solver FastHenry [2] to compute R and ωL for a typical global interconnect, which is $0.8\mu m$ wide, $2\mu m$ tall, and $2000\mu m$ long. One may easily see that ωL starts to outweigh the resistance at the operating frequency of approximate one gigahertz. As the operating frequency

is larger than the clock frequency due to the harmonic effect,[1] on-chip inductance should be considered in the layout design for circuits of gigahertz clock frequencies.

Furthermore, we compare the self inductance and mutual inductance in Figure 1(b). We designed an eighteen-bit signal bus sandwiched between two coplanar power/ground nets, where all wires are $0.8\mu m$ wide, $2\mu m$ tall, $2000\mu m$ long, and are separated by $0.8\mu m$. We computed the loop inductance for these wires using FastHenry. In Figure 1(b), the leftmost data-point stands for the self inductance of the leftmost signal net, and the rest of the data-points are mutual inductance between the leftmost signal net and the remainder of the signal nets. Clearly, the inductance has a "long-range" effect in the sense that the mutual inductance between non-adjacent nets cannot be ignored when compared to the self inductance. Note that the capacitance is a "short-range" effect in the sense that the mutual capacitance between non-adjacent nets is negligible, so that interconnect modeling and design under the RC model needs to consider only the net under study and its two adjacent nets, or in most cases, to consider just the net under study by assuming the worst-case mutual capacitance to the adjacent nets. This single-net based approach no longer works for the RLC model due to the long-range inductive effect. Hence, interconnect modeling and design under the RLC model should consider multiple coupled nets simultaneously, and is *inherently* more difficult compared to interconnect modeling and design under the RC model.

The remainder of this chapter is organized as follows: In Section 2, we present an efficient approach to compute inductance from the interconnect geometry, and use the resulting inductance model and SPICE simulation to evaluate the impact of layout optimization techniques such as wire sizing and spacing, buffer insertion and shield insertion. In Section 3, we describe the formulation and algorithm for the simultaneous shield insertion and net ordering problem. Discussions about recent progress are included in each section.

---

[1] The operating frequency is decided by the signal rise time t_r. The knee frequency can be defined as $F_{knee} = 0.5/t_r$ and be used as the operating frequency to compare ωL and R, as "the behavior of a circuit at (operating) frequencies above F_{knee} hardly affects digital performance" [3]. Similar conclusion was drawn in [4] using the concept "significant frequency". More sophisticated rules to judge the significance of inductance can be found in [5, 6, 7].

2 Inductance Extraction

Inductance extraction computes inductance from complex 3-dimensional (3D) interconnect structures. It is considered as a very difficult problem, because the inductance in essence is defined for the current loop but the current return path is not well defined for on-chip interconnects. The key idea to solve the problem is the *partial element equivalent circuit (PEEC)* model. It is developed in [8] and was introduced to the circuit design field in [9, 10]. The inductance in the PEEC model, in short, the *partial inductance* for a wire segment is defined as the portion of the loop inductance for the wire segment, assuming the wire segment forms a loop with the infinity. Numerical field solvers have been developed based on the PEEC model [10, 2, 11].

However, extraction of parasitic parameters (resistance, capacitance and inductance) via numerical field solvers is hard to support during iterative procedures of simulation and optimization for on-chip interconnects. Accurate and efficient extractions of resistance and capacitance without using numerical methods have been achieved recently. Examples include the 2 1/2-D capacitance extraction methodology [12], and the statistically-based worst-case RC model [13]. Both are table-based approaches, and are suitable for iterative simulation and optimization purposes. In the following, we will first present two foundations concerning the inductance extraction. These two foundations are based on the PEEC model, and will lead to an accurate and efficient table-based inductance extraction methodology. We then apply the table-based inductance model to compute the loop inductance for a coplanar waveguide to show the link between the loop inductance and partial inductance model, and apply the inductance model to study the impact of layout optimization.

2.1 Foundations for Inductance Extraction

There are multiple metal layers in a VLSI technology. We assume that wire traces on adjacent layers are orthogonal, and extract the inductance for a block, which contains n parallel traces (T_1, T_2,..., T_n) of same length on the same layer (see Figure 2). In addition, we also assume that the two most outside traces, T_1 and T_n, are dedicated power/ground traces. When the block size is three, it is a coplanar-waveguide, which is one of the three basic forms for transmission line, and is often used for clock tree in high-speed designs. When the block size is large, it models the bus structure

with outside power/ground traces that can be used for shielding only or for shielding and power supply at the same time. Because traces are orthogonal on adjacent layers, traces on layer $N+1$ and layer $N-1$ will not affect the inductance of traces on the current layer N [4].

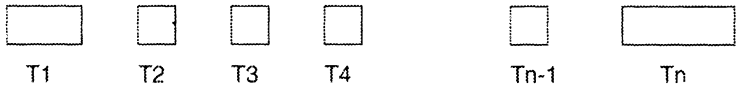
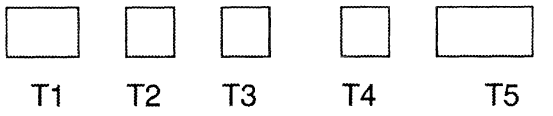


Figure 2: The cross-section view for a block of n traces, where T_1 and T_n are dedicated power/ground traces. The widths for the traces are W_1, W_2, ..., and W_n, respectively. The spacings between them are S_1, S_2, ... and S_{n-1}, respectively.

Note that the capacitive effect is a "short-range" effect in the sense that for a block, only the mutual capacitance between adjacent traces are important, and the rest of the mutual capacitance can be ignored. Therefore, for any trace, it is sufficient to solve the trace and its two adjacent traces via numerical extraction. In other words, we are able to reduce the n-trace capacitance problem to a number of 3-trace subproblems [12]. The inductive effect, however, is a "long-range" effect. For example, in Figure 3, we compute the inductance for a block with size n=5 by assuming that the wire thickness is $2.0\mu m$, wire width W1=$4\mu m$, W2=W3=W4=$0.8\mu m$, W5=$2\mu m$, all spacings are $0.8\mu m$, and the length is $4000\mu m$. We specify that T_1 and T_5 are power/ground traces that serve as current return paths, and run a 3D inductance tool RI3 in Raphael [11] to compute loop inductance. The result is the loop inductance in a 3x3 matrix, where current is assumed to return via the two power/ground traces T_1 and T_5. In the matrix, diagonal elements are self inductance, and off-diagonal elements are mutual inductance. The mutual inductance between T_2 and T_4 can not be ignored even though there is T_3 between them.

In general, there is a significant mutual inductance between any traces within a block (e.g., even for a block of size n=32). Due to this "long-range" effect, even though we assume that all signal traces have identical widths, and the spacings are identical, the brute-force way to build inductance tables will have large table sizes. The table for self inductance has six dimensions: two widths for power/ground traces, one width for signal traces, the trace location, and uniform spacing and length. Note that the trace length is



	T2	T3	T4
T2	1.73	1.15	0.53
T3	1.15	1.94	1.24
T4	0.53	1.24	1.92

Figure 3: Loop inductance (nH) by specifying that T_1 and T_5 are current return paths.

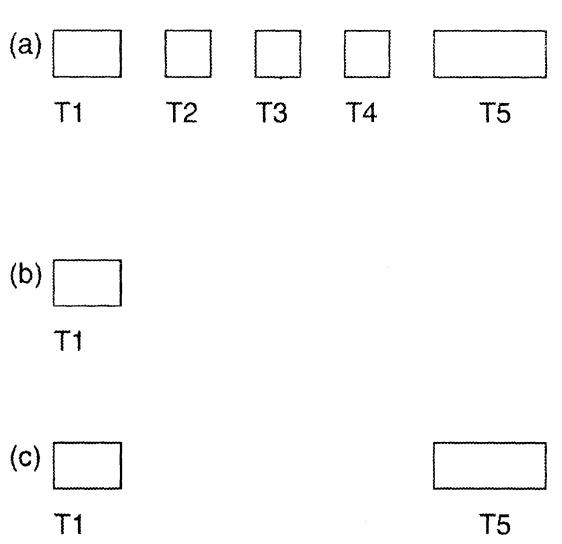
needed because the inductance is not a linear function of trace length. At the same time, the table for mutual inductance needs locations for two traces, which leads to seven-dimension tables. We may not afford to consider different widths and spacings for different traces.

Furthermore, the loop inductance in Figure 3 assumes that all current returns via the two power/ground traces, which may not be true for high frequency. For example, when only one trace is switching, its current may return from adjacent quiet traces. The right way to extract inductance for a block is to run RI3 [11] without specifying any power/ground traces as current return paths. Then, for the block in Figure 3, we obtain the partial inductance in a 5x5 matrix (see Figure 4(a)). Again, the diagonal elements are self inductance, and off-diagonal elements are mutual inductance. An important observation is that now the self inductance of a trace depends only on the trace itself, and the mutual inductance of two traces depends only on the two traces themselves. For example, in Figure 4(b), we compute the self inductance L_{11} for T_1 with other traces removed, and obtain the same L_{11} as in Figure 4(a). In Figure 4(c), we compute the mutual inductance L_{15} for T_1 and T_5 with T_2, T_3 and T_4 removed, and obtain the same L_{15} as in Figure 4(a).

When we do not specify which traces are power/ground traces, we compute partial inductance (denoted as L_p) under the PEEC model. In general, we have the following foundations:

Foundation 2.1 *Self L_p of a trace is* solely *decided by the trace (its length, width and thickness).*

Foundation 2.2 *Mutual L_p of two traces is* solely *decided by the two traces (their lengths, widths and thicknesses, and the spacing between them).*



	T1	T2	T3	T4	T5
T1	6.17	5.43	5.12	4.89	4.66
T2	5.53	6.79	6.10	5.48	5.04
T3	5.12	6.10	6.79	6.10	5.33
T4	4.89	5.48	6.10	6.79	5.77
T5	4.66	5.04	5.33	5.77	6.50

	T1
T1	6.17

	T1	T5
T1	6.17	4.66
T5	4.66	6.50

Figure 4: Partial inductance (nH) without specifying any current return path: (a) a block of size $n = 5$, (b) only trace T_1 (with other traces removed), and (c) only two traces T_1 and T_5 (with traces T_2-T_4 removed).

2.2 Validation of Foundations

In order to validate the two foundations, the following illustrates the inductance extraction procedure under the PEEC model. The PEEC model was introduced in [9, 10], and has been widely used in numerical inductance extraction tools (for example, [11, 2]). Because the inductance is defined only for closed loops, the partial inductance of a trace can be viewed as the inductance of the trace as it forms a loop with infinity. If the current density is uniform in traces T_k and T_m, according to [10], the mutual inductance $L_{p_{km}}$ under the PEEC model is:

$$L_{p_{km}} = \frac{\mu}{4\pi}\frac{1}{a_k a_m}\int_{b_k}^{c_k}\int_{a_k}\int_{b_m}^{c_m}\int_{a_m}\frac{dl_k \cdot dl_m}{r_{km}} da_k \cdot da_m \qquad (1)$$

where a_k and a_m are cross-sectional areas, b_k and b_m are starting points, c_k and c_m are ending points, all for traces T_k and T_m, respectively. In addition, r_{km} is the distance between dl_k and dl_m, which represents differential

elements of length for traces T_k and T_m, respectively. When $k = m$, Eqn. (1) gives the self L_p of a trace.

In the case where the current is not uniform in a trace, the trace can be divided into rectangular filaments (see Figure 5). The current is assumed to flow along the length of each filament with a constant density within each filament. Therefore, Eqn. (1) may be used for each filament. It is easy to see that Foundations 2.1 and 2.2 hold for each filament with respect to Eqn. (1). I.e., the self L_p of a filament is *solely* decided by the filament, and the mutual L_p between two filaments is *solely* decided by the two filaments. The conclusions hold for cases of a single trace and multiple traces.

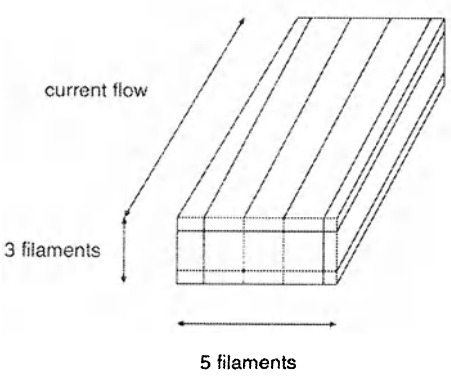


Figure 5: A trace is divided into 3×5 filaments. It is assumed that the current density is the same within a filament.

If we assume that trace T_k has P filaments, and trace T_m Q filaments, then $L_{p_{km}}$ is given by

$$L_{p_{km}} = \sum_{i=1}^{P} \sum_{j=1}^{Q} L_{p_{ij}} \tag{2}$$

where $L_{p_{ij}}$ is the mutual L_p between filament i of T_k and filament j of T_m. Again, when $k = m$, Eqn. (2) computes the self L_p for a trace. Because Foundations 2.1 and 2.2 hold for $L_{p_{ij}}$, it is easy to see that Foundations 2.1 and 2.2 still hold after using Eqn. (2) to compute L_p for traces.

2.3 Table-based Inductance Extraction

The two foundations enable us to reduce the n trace inductance problem into 1-trace subproblems to solve the self L_p, and into 2-trace subproblems to solve the mutual L_p. There is no loss of accuracy during the reduction. As given in [14], the self inductance may be solved by

$$L(nH) = 2l[\ln \frac{2l}{w+t} + 0.5 - k] \quad (3)$$

where $k = f(w,t)$ and $0 < k < 0.0025$, and l, w, and t are length, width and thickness of the trace in unit of cm. The mutual inductance for two traces of same width and length is

$$L(nH) = \frac{\mu_0 l}{2\pi}[\ln \frac{2l}{s} - 1 + \frac{s}{l}] \quad (4)$$

where s is the spacing between two traces, again in unit of cm.

length	$1000\mu m$		
width(μm)	1	1.2	2
Self L_p(nH)	1.480(0.0%)	1.461(-1.2%)	1.400(-5.4%)
Mutual L_p(nH)	1.101(0.0%)	1.100(-0.1%)	1.096(-0.5%)
length	$4000\mu m$		
width(μm)	1	1.2	2
Self L_p(nH)	7.028(0.0%)	6.951(-1.1%)	6.709(-4.5%)
Mutual L_p(nH)	5.551(0.0%)	5.508(-0.1%)	5.490(-1.1%)

Table 1: On-chip self and mutual inductance computed via numerical extraction for two parallel wire traces with pitch-spacing being $3\mu m$, and the thickness being $1\mu m$. For each length, three wire widths are assumed, namely, $1\mu m$, $1.2\mu m$ (i.e., the width has 20% variation compared to width=$1\mu m$), and $2\mu m$ (i.e., the wire is up-sized by 100% compared to width=$1\mu m$). Percentages of inductance changes with respect to inductance for width=$1\mu m$ are also given.

These equations give us two insights: First, the inductance for on-chip interconnects is not linearly scalable. Both self and mutual inductance are super-linear functions of the trace length. Secondly, because of the logarithmic operation of $\frac{2l}{w+t}$ and $\frac{l}{s}$, both mutual and self inductance is less sensitive

to variations of trace width and spacing as the capacitance and resistance are. The two insights are also verified by experiments with numerical inductance tools. For example, in Table 1, we report both self and mutual inductance for two parallel wire traces with pitch-spacing being $3\mu m$. For each length, three wire widths are assumed, namely, $1\mu m$, $1.2\mu m$ (i.e., the width has 20% variation compared to width=$1\mu m$), and $2\mu m$ (i.e., the wire is up-sized by 100% compared to width=$1\mu m$). One can see the following from the table: First, the inductance is not linearly scalable with respect to the wire length. For example, when the wire width is $1\mu m$ and the wire length increases from $1000\mu m$ to $4000\mu m$ (an increasing factor of 4x), the self inductance increases by a factor of 4.74x rather than 4x. Second, the inductance variation is only around 1% for the 20% variation of wire width. Therefore, there is no need to consider the impact of process variation for inductance extraction, even though the impact must be considered for resistance and capacitance extractions as in [13]. Finally, when we up-size the wire width by 100%, the inductance changes by up to 5.4% for self inductance, and by 4.5% for mutual inductance. Therefore, interconnect sizing and spacing might *not* be an effective way to change inductance[2].

There are limitations of applying the two equations however. First, they do not consider the skin depth and internal inductance; Second, widths are not considered for mutual inductance. Therefore, we will propose to build tables via numerical inductance extraction for self and mutual inductance.

There are two parts in the table-based inductance extraction. One is to pre-compute inductance tables. We assume that each layer has a nominal thickness, and build tables for different layers. The self inductance table has two dimensions: width and length. The mutual inductance table has four dimensions: widths for two traces, the spacing between them, and the length. The 3D inductance extraction tool RI3 [11] is invoked to solve a block of two traces for different combinations of lengths, widths, and spacings. The resulting self and mutual inductance is stored in tables. Note that only 2-trace subproblems need to be solved, because results to 1-trace subproblems are parts of results to 2-trace subproblems. In addition, the inductance depends on the skin depth, which is a function of frequency. We run RI3 [11] under the significant frequency. The significant frequency is defined as $0.17/t_r$, where t_r is the minimum rising/falling time [4]. It captures the

[2]On the other hand, because the coupling capacitance depends strongly on the spacing, the optimal interconnect sizing and spacing is effective to reduce RC delay (as well as capacitive coupling).

majority of high-frequency components for the signal with rising/falling time t_r.

The other part of the table-based inductance extraction is table lookup. For each trace in a block, we obtain a self inductance from tables for a given layer, length and width. For any combination of two traces T_i and T_j, we obtain a mutual inductance from tables for a given layer, widths, and spacing between T_i and T_j. A bicubic spline algorithm [15] will be used to compute inductance that is not given in the table.

2.4 Applications of Efficient Inductance Model

2.4.1 L_{eff} for Coplanar-waveguide

The coplanar-waveguide structure (a block of size $n = 3$, see Figure 6) is often used for on-chip clock trees in high-speed designs. Not to consider the inductive effect will lead to a significant underestimate of delay and noise. Therefore, the effective loop inductance (L_{eff}) of the signal trace needs to be computed in order to use the transmission line theory. In the following, we derive L_{eff} as a function of L_p for the three traces T_1, T_2 and T_3, where T_2 is the signal trace, and T_1 and T_3 are coplanar power/ground traces.

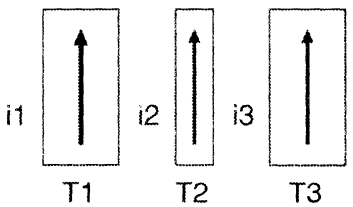


Figure 6: The top view of a coplanar-waveguide. T_1 and T_3 are dedicated power/ground traces.

L_{eff} is defined for the current loop that has two segments: the first segment is current i_2 through T_2; the second segment has two parallel branches, i.e., i_1 through T_1 and i_3 through T_3. According to the definition of L_{eff}, we have

$$\begin{aligned} \Delta V &= L_{eff}\frac{di_2}{dt} \qquad (5) \\ &= L_{p_{22}}\frac{di_2}{dt} + L_{p_{21}}\frac{di_1}{dt} + L_{p_{23}}\frac{di_3}{dt} - L_{p_{11}}\frac{di_1}{dt} - L_{p_{12}}\frac{di_2}{dt} - L_{p_{13}}\frac{di_3}{dt} \end{aligned}$$

and the two power/ground traces have the same voltage drop

$$L_{p_{11}}\frac{di_1}{dt}+L_{p_{12}}\frac{di_2}{dt}+L_{p_{13}}\frac{di_3}{dt} = L_{p_{13}}\frac{di_1}{dt}+L_{p_{23}}\frac{di_2}{dt}+L_{p_{33}}\frac{di_3}{dt} \quad (6)$$

finally, the current is conservative according to KCL

$$i_1+i_2+i_3 = 0 \quad (7)$$

L_{eff} can be derived by simultaneously solving (5)-(7). When T_1 and T_3 are symmetric with respect to T_2, L_{eff} is

$$L_{eff} = L_{p_{22}}-2L_{p_{23}}+\frac{L_{p_{11}}}{2}+\frac{L_{p_{13}}}{2} \quad (8)$$

Experiments show that (8) using our partial inductance tables gives results almost identical to L_{eff} obtained by 3D extractions using RI3 [11]. For example, we compute the L_{eff} for a coplanar-waveguide structure with all three traces $10\mu m$ wide, $2000\mu m$ long, and separated by $2\mu m$. The L_{eff} given by our formula (8) using table-based partial inductance is 0.839nH, while L_{eff} given by RI3 is 0.840nH. The difference can be almost ignored.

2.4.2 Impact of Layout Optimization

We have shown that wire sizing and spacing are not effective to reduce the inductance. However, as wire sizing and spacing are effective to change the coupling capacitance, wire sizing and spacing are still effective to reduce the coupling noise between adjacent wires.

The above inductance model can be used to generate the RLC circuit model. In the following, we apply the resulting RLC model to study the impact of buffer insertion and shield insertion. We use a bus structure as an example. It has 18 signal traces, and two fat power-traces outside the signal traces. The wire thickness is $2.0\mu m$, and spacing is $0.8\mu m$, both for all traces. The width is $0.8\mu m$ for all signal traces, and is $16\mu m$ for two fat power-traces. We assume the following signal pattern: all signal traces are simultaneously switching up with rising time of 80ps, except that one of the two central signal traces is the quiet victim. We also assume that all devices, including drivers, buffers and receivers, are 40x of the minimum inverter in a representative $0.18\mu m$ CMOS technology. Based on SPICE simulations, we will show how buffer insertion and shielding insertion reduce the noise under the RLC model.

A. Buffer insertion

It is well known that buffer insertion is effective to reduce the RC delay and capacitive noise. It is also very effective to reduce the inductive noise, since the original longer current return loop becomes shorter, as the current returns through the inserted buffers. Furthermore, as we discussed in the above (see Table 1), the inductance is a super-linear function of the wire length, more precisely, the DC-connected wire length between devices such as drivers, buffers and receivers. Therefore, while inserting a buffer at the middle point of a long wire reduces the coupling capacitance by a half for each DC-connected wire, it reduces the mutual inductance by more than a half for each DC-connected wire.

In the following example via SPICE simulation under the RLC model, we assume that for all traces, the wire length is $4000\mu m$. We measure the noise at the far-end of the victim trace (the input node of receiver) for three cases: no buffer, one buffer, and three buffers inserted for each single trace, respectively. These buffers are inserted uniformly. Therefore, the DC-connected wire lengths are $4000\mu m$, $2000\mu m$ and $1000\mu m$ for three cases, respectively. As shown in Table 2, compared to the case of no buffer insertion, inserting one buffer reduces the noise by 21.1%, and inserting three buffers reduces the noise by 42.3%. Note that the noise is measured at the inputs of devices for the victim trace. Much less of noise will be observed if the measurement is made at the output of devices.

number of buffer inserted	0	1	3
wire length (μm)	4000	2000	1000
Noise (V)	0.71(0.0%)	0.56(-21.1%)	0.41(-42.3%)

Table 2: Comparison of noise between different buffer insertion solutions.

B. Shield insertion

A shielding trace is a wire directly (without through devices) connected to power or ground networks.[3] It can provide the dedicated current return

[3]The most convenient way to connect random shields is to add vias between shields and P/G wires in orthogonal routing layers. In this chapter, we assume that there is a connection between the P/G structure and every $200\mu m$-long shield.

path to reduce the inductive noise. In the following experiment, we assume that for all traces, the length is $4000\mu m$, and no buffers are inserted. We again measure the noise at the far-end of the victim trace (the input node of receiver) via SPICE simulation. We will insert shielding traces to make the far-end noise of the victim trace less than 0.25V.

When there is no shielding traces, the noise of victim trace is 0.71V. Then, we insert a shielding trace for every six signal traces, and increase the width W_s for shielding traces from $0.8\mu m$ to $2.4\mu m$. The noise is 0.22V with $W_s = 2.4\mu m$. Finally, we insert a shielding trace for every three traces. The noise is 0.17V when $W_s = 0.8\mu m$. As shown in Table 3, there is a clear trade-off between area and noise: we may reduce the noise by a factor of 4.2x while the total width, the sum of the total wire width and total spacing, of the bus structure is increased by 13%, and the total wire width is increased by 8.8%.

N_s	W_s	Noise (V)	total width (μm)	wire width (μm)
18	–	0.71	61.6	46.4
6	0.8	0.38	64.8	46.0
6	1.6	0.27	64.4	49.6
6	2.4	0.22	68.0	51.2
3	0.8	0.17	69.6	50.4

Table 3: Comparison of noise between different shielding insertion solutions. Column one (N_s) is the number of signal traces between two shielding traces, and column 2 (W_s) is the width for the shielding traces. Column 3 is the total width (including the total spacing) of the the bus structure, and column 4 is the total wire width.

2.5 Conclusions and Discussions

In this section, we have present an efficient table-based inductance model for parallel coplanar wires. Several papers have extended this model. Our table-based inductance model is not applicable to strip lines and micro striplines with power/ground planes. In [16], two new foundations have been developed to compute the loop inductance for strip lines and micro striplines, in a fashion similar to Foundations 2.1 and 2.2. Our table-based inductance model is not applicable to random wires either. Different from parallel wires

that have the same lengths, and are aligned and routed in the same metal layer, random wires may have different lengths, and may routed in the different layers. In [17, 18], formulae have been developed to compute the inductance for random wires based on the table-based inductance model for parallel wires.

Further, we have studied the impact of layout optimization based on SPICE simulation using the RLC circuit model. We show that both buffer insertion and shield insertion are effective to reduce noise. Related studies and tutorials can be found in [19, 4, 20, 21]. In the next section, we will discuss algorithms to leverage the shield insertion with simultaneous net ordering.

3 Simultaneous Shield Insertion and Net Ordering

Interconnect synthesis has been extensively studied for RC interconnect models, including routing topology generation, wire sizing, wire spacing, device sizing, buffer insertion, net ordering, and combinations of these design components. A comprehensive survey of these work can be found in [1].

There has been some preliminary work using the RLC interconnect models as well, including routing topology generation [22], wire sizing [23] and buffer insertion [24]. However, all of them assume a single RLC net, which is not adequate as discussed in Section 1. Below, we present the formulation and algorithm for simultaneous shield insertion and net ordering (SINO) problem. It is one of the earliest work that studies the interconnect synthesis for multiple RLC nets.

3.1 Preliminaries

A. Coplanar Interconnect Structures

Again, we consider only parallel coplanar interconnect structures with all wires having the same lengths. These are characterized as a number of signal traces and power/ground traces which run parallel in the same layer. We give an example of this structure in Figure 7. In the figure, P and G represent the power and ground grids (P/G grids), s represents signal wires (denoted as s-wires), and g is a shield wire connected to P/G grids. Both P/G grids and shield wires provide dedicated current return paths

for signals, and are denoted as g-wires in this chapter. We use the terms, "trace", "wire" and "net" interchangeably.

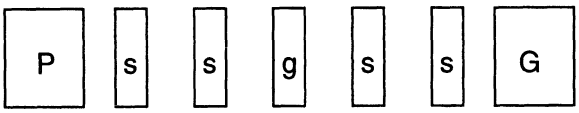


Figure 7: A cross-section view of a coplanar interconnect structure with a shield inserted.

A coplanar interconnect structure can be represented by a string, where each symbol stands for an s-wire or a g-wire. For example, the interconnect structure in Figure 7 can be represented by $gssgssg$ if we do not distinguish these s-wires (or alternatively, g2sg2sg). If we label the s-wires from left to right as s_1, s_2, s_3 and s_4, then the string $gs_1s_2gs_3s_4g$ is a unique representation of a net ordering and shield insertion solution (referred to as a SINO solution or a SINO string). In this chapter, a SINO string implicitly includes a shield trace (the power and ground grids) as its first and last element. These P/G grids are shield resources, but are not considered in solution size computations as they are present generally, with or without noise considerations. The "size" of a SINO solution can be determined directly from the length of the SINO string. As an example, consider the following: $< g > s_1s_3gs_2s_4s_5gs_7s_6s_0 < g >$. This string represents an eight (signal) bit interconnect structure with two g-wires (plus two implicit g-wires for the P/G grids denoted as $< g >$). Note that we can apply our formulations and algorithms to be presented to any group of wires which may contain pre-routed g-wires more than just a pair of P/G grids. We call the group of wires sandwiched between adjacent g-wires a block, and the number of s-wires in a block as the block size. A block can be represented as a substring of a SINO string (i.e. block 0 of the above string would be written as s_1s_3). As in the original SINO string, the g-wires on each end are implicit with the substring.

B. Characteristics and Model of Inductive Coupling

We illustrate the characteristics of inductive coupling in Figure 8, where we show the mutual inductance from the leftmost s-wire to all other s-wires. Cases (a) and (b) in the figure show two interconnect structures, $g18sg$ and

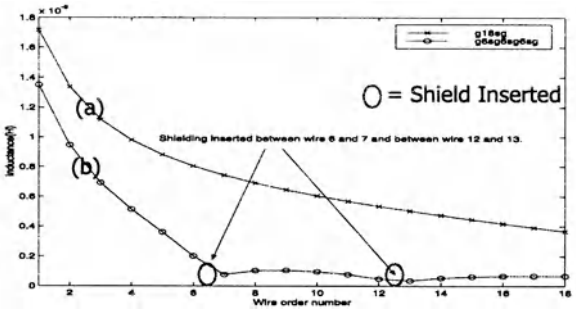


Figure 8: Illustration of mutual inductive coupling from a signal wire (s_1) to other signal traces for two coplanar structures. Structures (a) and (b) show coupling from s_1 to all other s-wires as a function of wire-order number for the g18sg and g6sg6sg6sg structures, respectively.

g6sg6sg6sg, respectively. As shown by (a) where there is no shield between s-wires, the mutual inductance decreases slowly from left to right. This implies that net ordering is not effective to reduce inductive coupling. In comparing (a) with (b) in the figure, the mutual inductance between wires separated by shields becomes much smaller compared with coupling to wires within the same block. Therefore, shields are effective to reduce inductive coupling.

In order to efficiently model inductive coupling between two arbitrary s-wires, we use the coupling coefficient between two s-wires as the figure of merit. The coupling coefficient between two nets s_i and s_j is defined as

$$K_{ij} = \frac{m_{ij}}{\sqrt{l_i \cdot l_j}}$$

where m_{ij} is the mutual inductance between s_i and s_j, and l_i and l_j is self inductance for s_i and s_j, respectively. We use an "effective coupling model", i.e. K_{eff} model, in this chapter in order to efficiently compute the coupling coefficient. In this model, when nets i and j are in different blocks, the coupling coefficient is $K_{ij} = 0$.[4] When the two nets are in the same block, as shown in Figure 9 where N_i and N_j are track ordering numbers for the two s-wires, and g_l and g_r are track ordering numbers for the two edge

[4]This assumption may occasionally lead to under-estimating inductive coupling, which will be discussed in Section 3.5.

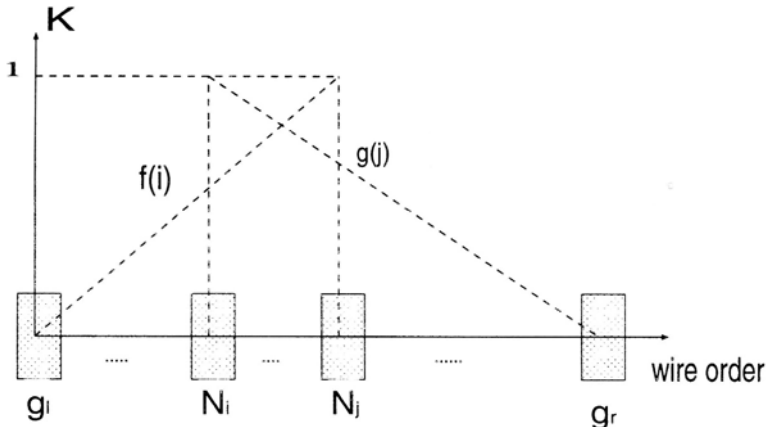


Figure 9: Illustration of K_{eff} computation. N_i and N_j are two signal wires in the same block sandwiched by ground wires g_l and g_r. $f(i)$ and $g(j)$ are two linear functions of the wire order number i, j as shown by the sloping dotted line. The mutual inductance coupling is given by the mean of $f(i)$ and $g(j)$.

g-wires, the coupling coefficient is computed as

$$K_{i,j} = \frac{(f(i) + g(j))}{2}$$

where $f(i)$ and $g(j)$ are two functions defined as follows: We assume that the function f is 0 at g_l, and is 1 at N_j; therefore $f(i)$ is the linear interpolation, i.e., $f(i) = \frac{(N_i - g_l)}{(N_j - g_l)}$. Similarly, we can compute $g(j) = \frac{(g_r - N_j)}{(g_r - N_i)}$.

The K_{eff} model is independent of the width, length and spacing of s-wires and g-wires. As an illustration in Figure 10, we compare the coupling coefficients given by the K_{eff} model and the three-dimensional field solver FastHenry [2]. In the figure, the x-axis indicates the coupling coefficient given by FastHenry and the y-axis indicates the coupling coefficient given by the k_{eff} model. We use coplanar interconnect structures $g(6sg)^2 6sg$ (i.e., $g6sg6sg6sg$) and $g(3sg)^5 3sg$. The wire width is $0.8\mu m$, length is $1000\mu m$, thickness is $2\mu m$ and the spacing between wires is $0.8\mu m$. The three dashed lines in Figure 10 indicate differences of 0%, +15% and −15%, respectively. One can easily see that most of the data points lie within the ±15% error range.

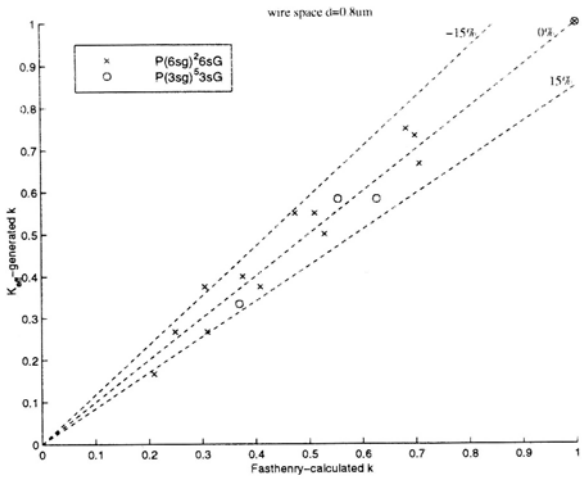


Figure 10: Accuracy of K_{eff} model. The x-axis is the mutual inductance coefficient as computed by FastHenry. The y-axis is the K_i value calculated by the K_{eff} model. Error lines for +15%, 0%, and -15% difference between the models indicate the quality of the K_{eff} model.

C. Net Sensitivity

We define two nets s_1 and s_2 to be sensitive to each other if a switching signal on s_1 will cause s_2 to malfunction (due to extraordinary crosstalk or delay variation) or vice-versa. Net sensitivity is depicted graphically in Figure 11.

The sensitivity for all s-wires in a given problem can be represented compactly with a sensitivity matrix S of size $n \times n$ (where n is the number of s-wires) as shown in Figure 12. The graphical representation of the sensitivity matrix (essentially an undirected graph structure), is also shown in Figure 12. An entry of 1,0 in location (i, j) indicates that s_i and s_j are sensitive or not sensitive, respectively, to one another. By definition, the matrix must be symmetric since the underlying graph is undirected (i.e. $S_{ij} = S_{ji}$). For all formulations, we assume that an appropriate sensitivity matrix indicating design parameters and net relationship semantics is given a priori.

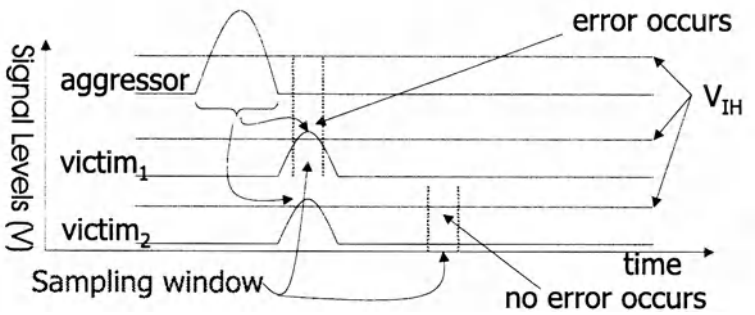


Figure 11: Illustration of net sensitivity. The y-axis indicates signal (voltage) level and the x-axis indicates time. The switching event on the aggressor induces a noise voltage in the two victims as shown. For $victim_1$, the noise pulse occurs during a sampling window–hence the aggressor and $victim_1$ are sensitive. For $victim_2$, the noise pulse does not occur during a sampling window–hence the aggressor and $victim_2$ are not sensitive.

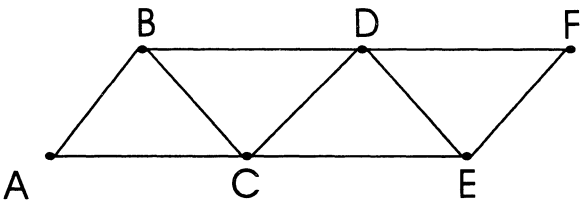
3.2 Formulation of Optimal SINO Problems

We say that an s-wire s_i is capacitive noise free if it is not directly adjacent to any other s-wire s_j that is sensitive to it. Similarly, we say that s_i is inductive noise free if it does not share a block with any other sensitive wire s_j. We say a placement P (or equivalently, a SINO solution, or a SINO string) is noise free if, and only if, all nets s_i within P are free of both capacitive noise and inductive noise. With respect to these concepts, we define the following SINO problem to eliminate Cx and Lx noise and call it the noise free SINO problem (SINO/NF).

Formulation 3.1 *(Optimal SINO/NF problem) For a given placement P, find a new placement P' by simultaneous shield insertion and net re-ordering such that P' is noise free and the total area of P' is minimal.*

In general, the SINO/NF problem is over-constrained and may lead to over-designed solutions as shown in Section 3.5. To address more realistic design constraints, we define the following SINO problem to meet a given noise bound and call it the noise bounded SINO problem (SINO/NB).

Formulation 3.2 *(Optimal SINO/NB problem) For a given placement P, find a placement P' with the minimum area by simultaneous shield insertion*



	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	1	0	0
C	1	1	0	1	1	0
D	0	1	1	0	1	1
E	0	0	1	1	0	1
F	0	0	0	1	1	0

Figure 12: Illustration of a sensitivity graph and the corresponding sensitivity matrix. The nodes in the graph correspond to signal nets, with an arc between the nodes indicating that two nets are sensitive to one-another. The sensitivity matrix is an equivalent representation for the sensitivity graph shown.

and net re-ordering such that any s_i in P' is free of capacitive noise and its inductive coupling to all sensitive wires s_j is less than a given value.

In this chapter, we use the K_{eff} model as the figure of merit for inductive coupling. If s-wire s_i is a net sensitive to s-wire s_j, we may compute the total amount of inductive coupling for s_i as:

$$K_i = \sum_{j \neq i} S_{ij} \cdot K_{ij}$$

Therefore, we solve the following optimal SINO/NB-K_{eff} problem: For a given placement P, find a new placement P' by simultaneous shield insertion and net re-ordering such that P' is free of capacitive coupling and the inductive coupling K_i satisfies $K_i \leq k_i$ for all K_i where k_i is given a priori and is a measure of inductive noise that can be tolerated in an s-wire to maintain correct operation. Throughout the rest of this chapter, we use a uniform value of k_i denoted as K_{thresh} (a noise threshold for K). However, our formulation, algorithms, and implementation are all able to handle non-uniform k_{thresh}. Note that we only consider insertion of minimum width

shields because it has already been shown that, in general, using additional shields is more effective than increasing the shield wire width.

We attempt to solve both the SINO/NF problem and SINO/NB-K_{eff} problem in Section 3.3. For simplicity of presentation, we use SINO/NB as shorthand for SINO/NB-K_{eff} throughout the remainder of the chapter. Note that our formulation and algorithms to be presented are applicable to inductive noise models more accurate than the K_{eff} model.

3.3 SINO Properties

Theorem 3.3 *The Optimal SINO/NF problem is NP-hard.*

Theorem 3.4 *The optimal SINO/NB problem is NP-hard.*

Given that the SINO/NF and SINO/NB problems are NP-hard, we will focus on developing heuristic/approximate algorithms to solve the problems with satisfactory results. Before we present these algorithms, we first introduce the following lower bound for the optimal SINO/NF solution:

Theorem 3.5 *The maximum clique size in the sensitivity graph is a lower bound of the number of blocks required in all optimal SINO/NF solutions.*

The theorem follows directly from the formulation of the sensitivity graph and the definition of the maximum clique in a graph [25]. To illustrate this conclusion and further show that the maximum clique size is not an upper bound for the total number of blocks in optimal SINO/NF solutions, we present two examples in Figure 13. For the sensitivity graph in Figure 13(a), it is easy to verify that the graph has a maximum clique size of two. An optimal SINO/NF solution is $ABgCDE$, and there are two blocks in the solution. The sensitivity graph in Figure 13(b) is nearly the same as in Figure 13(a), except that an edge is added between C and D, i.e. nets C and D are sensitive to each other. The reader may easily verify that indeed this graph still only has a maximum clique size of two, but it is not possible to find a SINO/NF solution for this graph with two blocks. One optimal solution, consisting of three blocks, is $AEgCDgB$. Therefore, the maximum clique size is not a strict upper bound of the number of shields required in an optimal SINO/NF solution. Comparisons between the lower bound and the number of shield wires used will be presented in Section 3.5 to illustrate the quality of our approximation algorithms.

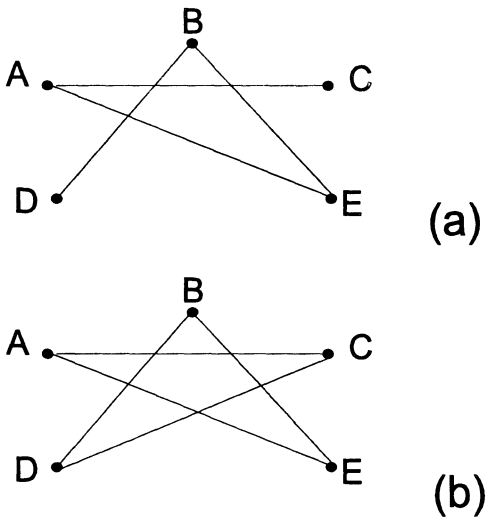


Figure 13: Sensitivity graphs: (a) Both the maximum clique size and the number of blocks are two; (b) The maximum clique size is two, but the minimum number of required blocks is three.

3.4 SINO Algorithms

We develop four approximate algorithms for solving the SINO/NB problem: greedy-based shield insertion (SI) algorithm, net ordering for minimizing Cx noise followed by SI algorithm (NO+SI algorithm), graph-coloring based SINO algorithm (SINO/GC algorithm), and simulated-annealing based SINO algorithm (SINO/SA algorithm). We solve the SINO/NF problem by using the SINO algorithms and setting the noise bound to zero. We will compare different problem formulations and algorithms in Section 3.5. We consider Lx and Cx noise explicitly in all algorithms. In order to more succinctly describe the algorithms and not clutter them with trivial details, we also define the following operations and quantities:

- *Insert_Shield*(a): Given a placement P of size m, move all wires (s-wires and g-wires) at locations $a, a+1, ..., m-1$ to locations $a+1, a+2, ..., m$ in the placement, creating a new placement P'. Then, insert a g-wire at location a in P'. Finally set $P = P'$.

- *Compute_Coupling*(s_i): Given a placement P of size m, for each $s_k \neq$

s_i in the current block in P, if s_i is sensitive to s_k compute the K_{eff} between s_i and s_k. Sum the K_{eff} over all s_k.

- *Compute_Block_Coupling*(s_i): For each s_i in the current block, find the maximal K_{eff} computed by *Compute_Coupling*(s_i) and return it (the maximal K_i for any s-wire within the block).

- *Max_Clique*(S): Compute the maximum clique in sensitivity graph S.

- *Compute_Placement_Cost*(): Compute the cost for a placement. Details of this are given in Section 3.4.

With these definitions in place, we can succinctly describe our SINO/NB algorithms. The SI and SINO/GC algorithms can be described easily and intuitively. SINO/SA is slightly more complicated, hence we assume that the reader is familiar with SA (for a discussion of SA in other VLSI design contexts, see [26], [27]).

A. Greedy Based Shield Insertion Algorithm

In Figure 14, we present the greedy-based shield insertion (SI) algorithm. The essence of the algorithm is the following: Run through the given placement P. If we encounter two adjacent sensitive nets, insert a shield wire between them. Also, at each location in the placement, calculate the maximum value of K_i that would exist in the current block if we allowed net s_i to become a member. If K_i is greater than K_{thresh}, then create a new block.

```
SI Algorithm: Given a placement P
for each s-wire s_i in P at location a:
        if s_j at location P(a − 1) is sensitive to s_i
                Insert_Shield(a)
        BC = Compute_Block_Coupling(s_i)
        if (BC > K_thresh)
                Insert_Shield(a)
                P(a + 1) = s_i
endfor
```

Figure 14: Greedy shield insertion algorithm (SI)

Because one shield wire is needed for every pair of adjacent sensitive wires, the solution given by the SI algorithm depends on the initial placement. Obviously, the number of shield wires can be reduced by first running existing net ordering algorithms to re-order nets so that no sensitive nets are adjacent to each other, then invoking the SI algorithm. This leads to the NO+SI algorithm.

B. Graph-Coloring Based SINO Algorithm

```
Graph Coloring (GC) Algorithm:
Given an initial sensitivity graph S of signal nets:
MC = Max_Clique(S)
P = new placement with MC blocks
for each s-wire s_i
    for each block b in P (sorted from largest to smallest)
        if Compute_Block_Coupling(s_i) == 0
            Insert s_i into b
            next s_i
    endfor
    for each block b in P
        if Compute_Block_Coupling(s_i) < K_thresh
            Insert s_i into b (adjacent to non-sensitive nets)
            next s_i
    endfor
    Insert_Shield(end_of_placement)
    Insert s_i into the new block
endfor
```

Figure 15: Graph-Coloring SINO Algorithm (SINO/GC)

In Figure 15, we present the graph-coloring based SINO (SINO/GC) algorithm. This algorithm attempts to approximate a solution to the weighted graph coloring problem by using the maximum clique as a starting point. It works in the following way: First, determine the maximum clique in the sensitivity graph S. We have shown that this is a lower bound for the number of shields required in the SINO/NF problem. Let m be the maximum clique size. We first create a placement with m blocks. We then take each net s_i and try to place it into a block with no other sensitive wires. If we cannot do this, we insert s_i into a block where inserting s_i will cause the

least amount of noise to be added but without causing a noise violation. If we cannot place s_i without creating a K_{thresh} violation, simply create a new block and place s_i into it.

Note that the SINO/NF problem can be mapped into the graph-coloring problem as outlined in the proof of Theorem 3.5, and then be solved using the graph coloring algorithms given in [28], [29].

B. Simulated Annealing Based SINO Algorithm

In Figure 16, we present the simulated-annealing based SINO (SINO/SA) algorithm. We give the details of our SINO/SA algorithm in the following subsections:

Cost Function *Compute_Cost*(P) computes the cost for a placement P. The cost is the weighted sum of the following components: (i) *Cap_violation*: total number of nets that are adjacent to their sensitive nets in P; (ii) *Area*: total number of g-wires present in P; (iii) *Ind_Noise*: total number of $K_i > K_{thresh}$ violations in P; and (iv) *Inductance Violation Figure*. It is computed for a placement as shown in Figure 17. The purpose of the Inductance Violation Figure is to penalize a placement for the magnitude of K_{thresh} violations. Its usage (as opposed to simply forbidding placements P' that have K_{thresh} violations) allows the algorithm to potentially trade inductive noise violations for smaller overall placement size depending on the result desired, and can be useful in different SINO formulations. The weighting factor for each cost component can be tuned for different design objectives. In this chapter our stated goal is to minimize placement size without violating K_{thresh} noise constraints, hence weighting factors were chosen to help us achieve these goals with maximal efficiency.

Random Moves *Random_Move*(P, P') performs one of the following changes to placement P to generate a new placement P': (i) Combine two random blocks in P, (ii) Swap two random s-wires in P, (iii) Move a single random s-wire to a new and random location, (iv) Insert a g-wire at a random location in P. It is worthwhile to note that combining two random blocks in a placement P is also equivalent to removing a g-wire if the two blocks are adjacent. Moves which create two adjacent g-wires in a placement are categorically rejected and a new move is tried.

```
Simulated Annealing Algorithm: Given a placement P:
Repeat
        Temp = Initial_Temperature;
        Repeat
                Random_Move(P, P');
                Candidate_Cost = Compute_Cost(P');
                ds = Candidate_Cost - Compute_Cost(P);
                if (ds < 0)
                        P = P';
                else
                        r = RANDOM(0, 1);
                        if (r < exp(-ds/Temp))
                                P = P';
        Until equilibrium at Temp is reached;
        Temp = Temp * Temperature_Adjustment;
        /*(0 < Temperature_Adjustment < 1)*/
Until Temp == Freezing_Point;
```

Figure 16: Simulated Annealing SINO Algorithm (SINO/SA)

```
Total_Violation_Figure = 0;
for each s_i in placement P
        if Compute_Coupling(s_i) > K_thresh
                Total_Violation_Figure+ = (1 + K_eff - K_thresh)^3 - 1
end
```

Figure 17: Computation of the Inductance_Violation_Figure

Temperature Adjustment and Stopping Criterion The method of temperature adjustment is shown in Figure 16. We use a simple multiplicative constant of the current temperature. At each temperature step, the variance of the current placement cost from its previous value is taken and averaged over several random moves to determine the stability of the system at each temperature. When the variance is less than a set threshold, we move to the next temperature step. The starting temperature, freezing point, temperature adjustment, and variance threshold factors were all determined experimentally.

3.5 Experimental Results

	SINO/NF	SINO/NB			
K_{thresh}	GC	SI	NO+SI	GC	SA
Net Sensitivity Rate: 10%					
0.5	3.5(2.0)	8.0	4.9	2.6	2.2
1.0		6.1	3.3	2.2	2.0
1.5		5.1	2.2	2.0	1.0
2.0		5.0	1.8	2.0	1.0
Net Sensitivity Rate: 30%					
0.5	6.5(4.0)	13.4	6.9	5.0	4.5
1.0		12.6	4.8	4.2	3.8
1.5		12.1	3.6	4.0	2.9
2.0		11.9	3.0	4.0	2.0
Net Sensitivity Rate: 60%					
0.5	12.0(9.0)	22.2	9.0	9.9	7.9
1.0		22.1	6.0	9.0	5.8
1.5		22.0	4.9	9.0	4.8
2.0		22.0	4.0	9.0	4.0

Table 4: Number of shields inserted by SINO algorithms. The numbers in parenthesis are lower bounds on the number of shields required in SINO/NF solutions.

We have implemented all algorithms in the C programming language, and have tested our implementations using a large number of examples. In this section, we first compare results obtained by different approximate algorithms to the SINO/NB formulation, and then compare results given by two formulations (SINO/NF versus SINO/NB). We also report the running time for different formulations and algorithms.

We use a coplanar interconnect structure containing 32 s-wires as an example to determine the performance of the algorithms for different combinations of K_{thresh} and sensitivity rate. We consider the following values for K_{thresh}: 0.5, 1.0, 1.5 and 2.0. When $K_{thresh} = 0.5$, the total of coupling coefficients for a net is less than 0.5 in the target SINO solution. We iterate the following sensitivity rates: 10%, 30% and 60%. When the sensitivity rate is 10%, each net is sensitive to 10% of all nets, and these sensitive nets are picked randomly for the given s-wire. The number in parenthesis is the SINO/NF lower bound on the number of shields.

For each combination of K_{thresh} and sensitivity rate (see Table 4), we report the resulting number of shields for different formulations and algorithms. To make the comparisons "fair" among the different algorithms, we run each algorithm on the same initial placement for 20 different random placements/sensitivities. The average of these 20 runs is shown in Table 4. Note that there is no entry in the tables for Cx noise because there was not any in all cases. Finally, it is worthwhile to point out that we did not tune our SINO/SA algorithm for different examples.

3.5.1 Comparison between Different SINO/NB Algorithms

We compare the following approximate SINO/NB solutions given by the following algorithms: greedy-based shield insertion (SI), net ordering followed by SI (NO+SI), graph-coloring based SINO (SINO/GC), and simulated-annealing based SINO (SINO/SA). One may easily see from Table 4: First, SI is always significantly worse than all of the other solutions in terms of the number of shields inserted. In the worst case, SI yields a result about 550% worse than SA in terms of the number of shields inserted (see sensitivity rate of 60% and K_{thresh} of 2.0). Furthermore, performing net ordering before SI, i.e., NO+SI significantly outperforms SI only because we need not insert shields for Cx violations which may be present without performing net ordering. On the other hand, at lower sensitivity rates (10% and 30%) and low K_{thresh} values, SINO/GC performs significantly better than NO+SI in terms of shields inserted (ranging from 46% better to 9% better) because SINO/GC can consider global placement of signal nets to minimize noise, whereas NO+SI cannot. As we move to higher sensitivity rate (60%) and K_{thresh}, we see that NO+SI begins to overtake SINO/GC because SINO/GC cannot generate solutions smaller than the maximum clique in the graph by design, hence not fully exploiting the noise bound. NO+SI also approaches the performance of SINO/SA under these same circumstances because in terms of probability, for high sensitivity rates and noise bounds, the initial random placement (used in NO+SI) is likely to distribute the many sensitive nets fairly uniformly giving less relative benefit from the flexibility to rearrange all nets and shields in SINO/SA. For a concrete illustration, at sensitivity rate of 60% and $K_{thresh} = 1.0$, NO+SI performs 33% better than SINO/GC and 4% worse than SINO/SA.

As we expect, SINO/SA always performs the best for any given setting. In terms of the number of shield wires, compared to NO+SI, its improvement increases as sensitivity rates and K_{thresh} decrease, as explained previously.

It is 4% better at sensitivity rate of 60% and $K_{thresh} = 1.0$, and is 55% better at sensitivity rate of 10% and $K_{thresh} = 0.5$. Therefore, it is important to consider simultaneous net ordering and shield insertion, rather than separated net ordering and shield insertion. The improvement of SINO/SA over SINO/GC does not depend much on the sensitivity rate, but increases when K_{thresh} goes up, for the same reason NO+SI outperforms SINO/GC at higher K_{thresh}. At sensitivity rate of 60%, it is 12% better for $K_{thresh} = 0.5$, and is 56% better for $K_{thresh} = 2.0$.

3.5.2 Comparison between SINO/NB and SINO/NF Formulations

We first compare the SINO/NF and SINO/NB solutions. We base our comparison on the results produced by the best SINO/NB algorithm (SINO/SA) and the GC algorithm for the SINO/NF formulation. For the smallest K_{thresh} value of 0.5 (in order to make SINO/NB most closely approximate SINO/NF), compared to the SINO/NF formulation, the SINO/NB formulation uses about 35% fewer shield wires for all sensitivity rates. Because our GC algorithm provides an approximate to the SINO/NF formulation, we computed the average lower bound of shield wires via average maximum clique size (based on Theorem 3.5), and present these lower bounds between parentheses in the column for GC algorithm, SINO/NF formulation. Compared to these lower bounds, the SINO/NB solutions still use up to 12% fewer shield wires for the lowest K_{thresh}=0.5.

	SINO/NF	SINO/NB			
	GC	SI	NO+SI	GC	SA
Runtime	0.1sec	0.1sec	0.1sec	0.3sec	1.5sec

Table 5: Approximate run times for SINO algorithms with sensitivity rate of 30%.

Finally, we report runtime in Table 5, where the times are for a single run for a single interconnect structure. Note that the running time for the NO+SI algorithm does not include the time for net ordering–we simply applied the SI algorithm to initial placements that are free of Cx noise. As verified in our experiments, whether the initial placement is free of Cx noise does not affect the quality of solution given by SINO/GC and SINO/SA algorithms. The machine used to collect running times has a 450MHz Intel Pentium II processor. All algorithms finished the test examples in a few

seconds. Therefore, large interconnect structures can be solved easily by formulations and algorithms we have proposed.

3.5.3 Fidelity of K_{eff} Model

The K_{eff} model is easy to compute and convenient to use at a higher design level or in an earlier design stage. However, it assumes that the current returns from the nearest shield, which is not true in general. The current often returns from quiet wires within the current block if there are plenty of quiet wires in this block. On the other hand, the current often returns from shields or quiet wires outside the current block when multiple wires in the current block switch simultaneously.

Despite the K_{eff} model is far away from accurate, we observe that it has a high fidelity in the following sense: an optimal SINO solution with a higher coupling value under the K_{eff} model also has a higher SPICE-computed noise over a distributed RLC circuit model using the partial inductance model. Our observation is based on the following experiment. We first generate a large number of optimal SINO solutions using our SINO algorithms, then rank these solutions by both the coupling value under K_{eff} model and SPICE-computed noise using an accurate RLC circuit model. We then compute the absolute difference between the two rankings for each optimal SINO solution. If the ranking difference is small enough, the K_{eff} model has a high fidelity. The similar ranking techniques have been used to study the fidelity of the Elmore delay model in [30, 31].

Further, we calculate the SPICE-computed noise difference with respect to the average ranking difference in the following way: let the average ranking difference be d. For a SINO/NB-v solution whose SPICE-computed noise ranking is i, we compute the relative difference between the $(i+d)$-th and i-th SPICE-computed noise, as well as that between the $(i-d)$-th and i-th SPICE-computed noise. Between the two values, the one with the larger absolute value divided by the i-th SPICE-computed noise value is defined as *noise difference* for the solution. In other words, it shows how much the noise difference is introduced if we use K_{eff} model to approximate the SPICE-computed noise voltage under an accurate RLC circuit model.

In our SPICE simulation, we generate a RLC segment for each $100\mu m$ wire segment, and a mutual inductance between any pair of two segments even though they belong to the same wire. We use the partial inductance model in [10, 32], without assuming any current return path. The parameters for the SPICE simulation of the detailed RLC circuit model are sum-

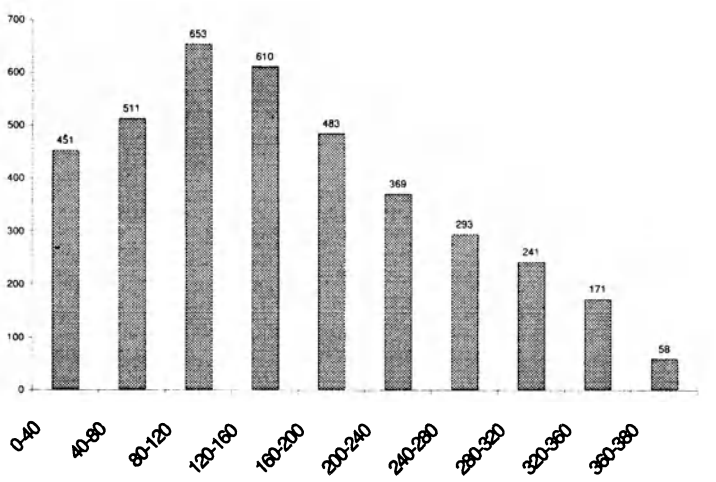


Figure 18: The distribution of the ranking difference for 3,840 SINO solutions.

marized in Table 6. We report in Figure 18 the distribution of the ranking differences for SINO solutions. The average ranking difference is only 109.387 over 3,840 SINO solutions (i.e., the relative error is 2.8486%). The noise difference is 5.3592% with respect to the average ranking difference.

Based on the above empirical evidence, we conclude that the K_{eff} model has a high fidelity over SPICE-computed noise using an accurate circuit model. Note that the above fidelity holds for optimal SINO solutions, because the SINO algorithms are able to distribute shields and quite wires evenly both spatially and temporally.

Vdd	clock	wire width	wire thickness	wire spacing	rising time	driver res	load cap
1.05V	3GHz	$1.0\mu m$	$1.1\mu m$	$0.8\mu m$	$33ps$	150Ω	60fF

Table 6: Experiment settings for fidelity study based on ITRS predicted $0.10\mu m$ technology

3.6 Conclusions and Discussions

In this section, we have formulated and solved the simultaneous shield insertion and net ordering (SINO) problem under the K_{eff} model. We have shown that the K_{eff} model has a high fidelity versus SPICE-computed noise

for SINO solutions.

In this chapter, we assume that the shields can be placed onto any track, without consideration of pre-routed P/G wires. In [33], a set of formulae has been developed to estimate the number of shields needed by optimal SINO solutions for given noise bound. Further, the simultaneous signal and power routing (SPR) problem has been formulated. The problem determines a regular P/G structure and finds a SINO solution with respect to the P/G structure such that the total routing area and number of shields are minimized. An efficient two-phase algorithm has been developed: a P/G structure is first speculated using the formula-based shield estimation, then the optimal SPR solution is found within the very limited neighborhood of the speculated P/G structure. Experiment has shown that the SPR solution reduces the total routing area by 1/3 compared to the min-area solution using a regular P/G structure without allowing any shields.

The K_{eff} model has been used in this chapter and [33]. A conservative RLC noise model has been developed in [34]. The noise model applies table-based models for capacitance and partial inductance, an inductance screening rule [35] to decide the scope of inductive coupling, and a five-pole model to compute noise voltage. Further, the SINO problem has been solved under this conservative model. SPICE simulations using an accurate RLC circuit model show that the new SINO algorithm leads to solutions with a smooth tradeoff between the number of shields and targeted noise bound.

4 Acknowledgment

The author would like to thank the following persons for valuable contributions to results presented in this chapter: Dr. Norman Chang, Dr. Shen Lin and Dr. Weize Xie at Apache Design Solutions, Dr. O. Sam Nakagawa at HP Labs, Mr. Kevin M. Lepak, and Mr. James Ma at University of Wisconsin.

A Web-based tool has been developed based on the inductance model presented in this chapter. This tool was developed by Mr. Min Xu at University of Wisconsin and is available at http://eda.ece.wisc.edu. New progress related to SINO and other interconnect synthesis techniques will be made available at the website too.

References

[1] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal*, vol. 21, pp. 1–94, 1996.

[2] M. Kamon, M. Tsuk, and J. White, "Fasthenry: a multipole-accelerated 3d inductance extraction program," *IEEE Trans. on MTT*, 1994.

[3] H. Johnson and M. Graham, *High-Speed Digital Design - A Handbook of Black Magic.* Prentice Hall, 1993.

[4] J. Lillis, C. Cheng, S. Lin, and N. Chang, *High-performance interconnect analysis and synthesis.* John Wiley, 1999.

[5] D. Zhou, F. P. Preparata, and S. M. Kang, "Interconnection delay in very high-speed VLSI," *IEEE Trans. on CAS*, July 1991.

[6] A. Deutsch *et al.*, "When are transmission-line effects important for on-chip wires," *IEEE Trans. on MTT*, 1997.

[7] Y. I. Ismail, E. G. Friedman, and J. L. Neves, "Figures of merit to characterize the importance of on-chip inductance," in *Proc. Design Automation Conf*, pp. 560–565, 1998.

[8] E. Rosa, "The self and mutual inductance of linear conductors," *Bulletin of the National Bureau of Standards*, pp. 301–344, 1908.

[9] A. Ruehli, "Inductance calculation in a complex integrated circuit environment," *IBM Journal of Res. and Dev.*, 1972.

[10] A. Ruehli, "Equivalent circuit models for three-dimensional multiconductor systems," *IEEE Trans. on MTT*, 1974.

[11] "Raphael user manual," *Avant! Corporation.*

[12] J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali, and S. H.-C. Yen, "Analysis and justification of a simple, practical 2 1/2-d capacitance extraction methodology," in *Proc. Design Automation Conf*, pp. 627–632, 1997.

[13] N. Chang, V. Kanevsky, O. S. Nakagawa, K. Rahmat, and S.-Y. Oh, "Fast generation of statistically-based worst-case modeling of on-chip interconnect," in *IEEE ICCD*, 1997.

[14] F. W. Grover, *Inductance Calculations: working formulas and tables.* Dover Publications.

[15] W. Press, S. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C.* Cambridge University Press, 1992.

[16] N. Chang, S. Lin,.L. He, O. S. Nakagawa, and W. Xie, "Clocktree RLC extraction with efficient inductance modeling," in *Design Automation and Test in Europe*, March 2000.

[17] X. Qi, G. Wang, Z. Yu, R. Dutton, T. Young, and N. Chang, "On-chip inductance modeling and RLC extraction of VLSI interconnects for circuit simulation," in *Proc. IEEE Custom Integrated Circuits Conference*, May 2000.

[18] M. Xu and L. He, "An efficient model for frequency-dependent on-chip inductance," in *Proc. the Sixth Great Lakes Symp. on VLSI*, 2001.

[19] M. W. Beattie and L. Pileggi, "IC analyses including extracted inductance model," in *Proc. Design Automation Conf*, 1999.

[20] L. He and S. Lin, "Interconnect modeling and design for gigascale systems-on-chip with consideration of inductance," in *IEEE International ASIC/SOC Conference*, 2000.

[21] Y. Cao, C. M. Hu, X. Huang, A. B. Kahng, S. Muddu, D. S. oobandt, and D. Sylvester, "Effects of global interconnect optimizations on performance estimation of deep submicron design," in *Proc. Int. Conf. on Computer Aided Design*, 2000.

[22] J. Cong and C.-K. Koh, "Interconnect layout optimization under higher-order RLC model," in *Proc. Int. Conf. on Computer Aided Design*, 1997.

[23] T. Xue, E. S. Kuh, and Q. Yu, "A sensitivity-based wiresizing approach to interconnect optimization of lossy transmission line topologies," in *Proc. IEEE Multi-Chip Module Conf.*, pp. 117–121, 1996.

[24] Y. I. Ismail and E. G. Friedman, "Effects of inductance on the propagation delay and repeater insertion in VLSI circuits," in *Proc. Design Automation Conf*, pp. 721–724, 1999.

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability.* W. H. Freeman, San Francisco, 1979.

[26] C. Sechen, "An improved simulated annealing algorithm for row-based placement," in *Proc. Int. Conf. on Computer Aided Design*, pp. 478–481, 1997.

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

[28] O. Coudert, "Exact coloring of real-life graphs is easy," in *Proc. Design Automation Conf*, 1997.

[29] D. Kirovski and M. Potkonjak, "Efficient coloring of a large spectrum of graphs," in *Proc. Design Automation Conf*, 1998.

[30] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Rectilinear Steiner trees with minimum Elmore delay," in *Proc. Design Automation Conf*, pp. 381–386, 1994.

[31] J. Cong and L. He, "Optimal wiresizing for interconnects with multiple sources," *ACM Trans. on Design Automation of Electronics Systems*, vol. 1, pp. 478–511, Oct. 1996.

[32] L. He, N. Chang, S. Lin, and O. S. Nakagawa, "An efficient inductance modeling for on-chip interconnects," in *Proc. IEEE Custom Integrated Circuits Conference*, pp. 457–460, May 1999.

[33] J. D. Ma and L. He, "Simultaneous signal and power routing under k_{eff} model," in *The 3rd Intl. Workshop on System-Level Interconnect Prediction*, 2001.

[34] K. M. Lepak, I. Luwandi, and L. He, "Simultaneous shield insertion and net ordering under explicit noise constraint," in *Proc. Design Automation Conf*, 2001.

[35] S. Lin, N. Chang, and O. S. Nakagawa, "Quick on-chip self- and mutual-inductance screen," in *International Symposium on Quality of Electronic Design*, March 2000.

Modeling and Characterization of IC Interconnects and Packagings for the Signal Integrity Verification of High-Performance VLSI Circuits

Yungseon Eo
Department of Electrical and Computer Engineering
Hanyang University, Ansan, Kyungki-Do, South Korea.
E-mail: eo@giga.hanyang.ac.kr

Contents

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 191–259.
© 2001 *Kluwer Academic Publishers.*

1 Introduction

Today's state of the art VLSI circuits, microwave integrated circuits, and next generation VLSI or ULSI circuit designs face new challenging problems, that is, signal integrity problems. With the advent of deep submicron semiconductor processing technology, there has been rapid development of IC with integration of more than several hundred million transistors. These VLSI circuits will be operated with clock frequencies greater than several GHz. However, such an increase of integration level and speed raises the risk of poor noise margins and timing malfunctions during circuit operations. Therefore, when designing high performance VLSI circuits, very accurate design methodologies are required.

One of the limiting factors of VLSI circuit designs is the signal integrity problem concerned with on-chip interconnects and package. As IC interconnects become narrower and are integrated in tighter physical configurations, they assume a pivotal role in high-speed circuit performance. Therefore, accurate characterization and modeling of the interconnect and package will be required in the high-speed circuit designs. Without accurate knowledge of their performance, it is impossible to precisely predict during a design stage the clock skew, crossstlak noise, signal delay, and waveform degradation (i.e., signal integrity verification). Thus, the risk of failure of high performance circuit design increases dramatically.

Accordingly, it is essential to simulate targeted circuits by predicting their important electrical design parameters. In complicated modern circuit

designs, a circuit without any simulation cannot be iterated upon quickly enough to get a profitable production to market. Simulation results should not only provide the exact electrical characteristics but also suggest design improvements to the circuit designer. Simulation necessarily requires models of the system characteristics and model parameters.

Note, while the characterization is used to extract model parameters and electrical characteristics of unknown systems experimentally, modeling is used to find the mathematical or physical expressions to explain and predict future experimental results. Thereby, it is possible through simulations to predict any system's characteristics by combining characterized and modeled components.

The chapter is organized as follows. First, on-chip interconnect modeling and its model parameter determination are presented. Next, signal delay models due to the interconnects are introduced, followed by the crosstalk noise. Finally signal integrity problem due to package(i.e., simultaneous switching noise) is explained.

2 Interconnect Modeling and Model Parameters

Electrical signal transmission phenomena on interconnect lines traditionally have been modeled using the Telegrapher's equations as follows

$$\frac{\partial V(x,t)}{\partial x} = -\left(RI(x,t) + L\frac{\partial I(x,t)}{\partial t}\right) \tag{1}$$

$$\frac{\partial I(x,t)}{\partial x} = -\left(GV(x,t) + C\frac{\partial V(x,t)}{\partial t}\right). \tag{2}$$

Their model parameters are per-unit-length-based four coefficient matrices, [R], [L], [C], and [G]. Thus, the four parameter matrices play a pivotal role in accurately predicting the electrical characteristics of interconnects. Ideally, the four model parameter matrices can be determined from semiconductor process information. This semiconductor process information includes the physical and electrical description of cross-sectional structures and their physical dimensions.

Although the full solutions of (1) and (2) give accurate signal variations, it takes a large amount of computation time to be used in complicated interconnect circuits. In VLSI circuit interconnects, the dielectric loss and inductive reactance can be neglected as a first approximation. Thus, interconnects have been previously modeled with RC ladder circuits, assuming finite RC charging time is a major delay source due to the interconnect lines

[1][2],[3][4]-[8]. The RC model can afford to be readily integrated into any existing CAD tools for signal integrity simulation. Thereby, with this RC model, many of the integrated circuit timing and noise verifications have been predicted during the layout and routing. However, although the RC model is simple as well as efficient, it is inaccurate as the circuit speed increases. That is, the RC charging time is only a part of the signal transient characteristics of IC interconnects. The importance of the inductance effect due to IC interconnects has been reported by many authors [9][10]-[12]. To overcome the inaccuracy problem of the RC model, an RLC model has been employed to verify the circuit timing and signal integrity [9][12]-[14]. However, the RLC model also has deficiencies in verifying the high-speed signal integrity since the frequency-variant transmission line parameters are not reflected in the RLC model. The waveform distortion and other delay sources due to the skin effect in the interconnect line and the interfacial polarization in the silicon substrate become increasingly important [15]-[17]. Since typical IC interconnects are fabricated on a lossy silicon substrate which operates as an imperfect ground plane, accurate signal variations on the IC interconnects cannot be investigated without considering the aforementioned effects. Early work in the IC interconnect modeling at microwave frequencies examined the microstrip on SiO_2-Si substrates and three possible modes of signal propagation were described [18][19]. Since then, the signal propagation in the transmission lines on semiconductor substrate has been analyzed by using various methods [20]-[23]. However, it is inherently difficult to simultaneously investigate both the silicon substrate effect and frequency-variant transmission line parameters with a circuit model. Due to such difficulties in the on-chip interconnect characterizations, s-parameter-based techniques have been employed which can implicitly include all these complicated parametric variations of such lines up to a broad frequency band. Up to date there have been many studies on experimental characterization since today's high-speed integrated circuit design requires accurate characterization of IC interconnects in several 10's of GHz frequency range by treating them as transmission lines [24]-[27]. The frequency variant transmission line parameters of the interconnect line on a SiO_2-Si substrate were experimentally extracted both in the frequency domain and in the time domain [24][25][27][28]-[30]. Since next generation VLSI circuits are expected to have lower noise margin due to lower power and higher speed operation, much tighter timing skew and noise budget are inevitable [31][32]. Thus, it will be necessary for the industry to characterize these interconnect lines directly from measurement and build a representative interconnect characterization database.

2.1 IC Interconnect Model Parameters

2.1.1 Interconnect Resistance (R) and Capacitance (C)

The simplest configuration of an IC interconnect is a single microstrip line on a silicon substrate. Metal interconnect lines in VLSI circuits occupy large amount of die area and are multi-layered. Metal layeres tend to be orthogonal to simplify design and reduce crosstalk between layers. Furthermore, metal layers will be similar to a solid conductor over a ground plane if spaces are smaller than a wave length of the signal transmitted. Thus, the upper level metal layers tend to be electromagnetically isolated from the substrate by the lower level metal conductors. Therefore, IC interconnect modeling can be divided into first level interconnect with lossy silicon substrate effects and upper level interconnect which is masked from substrate effects.

In RC models, the resistance, R, is the thin film resistance. Thus, the resistance can be readily determined by sheet resistance measurements. The resistance of line width(w), line thickness (t), and line length (l) is given by

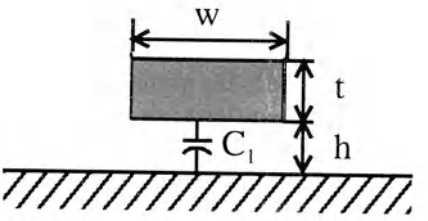
$$R = \rho\frac{l}{t \cdot w} = R_s\left(\frac{l}{w}\right) \tag{3}$$

where ρ and R_s are the resistivity and the sheet resistance, respectively. The resistance of the metal may have, in practice, large deviations from layout dimensions due to over etching or under etching. Therefore, considering the process variations on metals, the practical R may be bounded within the following inequality

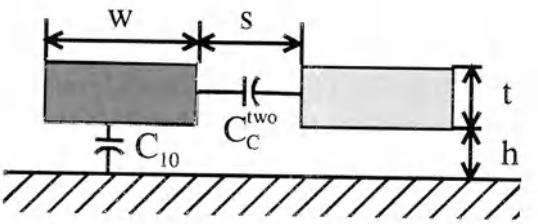
$$R_s\frac{l}{W_{max}} \leq R \leq R_s\frac{l}{W_{min}}. \tag{4}$$

Unlike the resistance, the capacitance is not so simply determined. Therefore, considerable efforts have been exerted on the capacitance determination [37][33][38]. Although numerous interconnect capacitance analysis and design methods have been developed for specific application purposes, the interconnect characterization approaches can be categorized as traditional full wave analysis, quasi-TEM analysis, the use of simulators, empirical models, low frequency measurements, and recent high-frequency-wave-based characterizations. Each technology has merits and demerits. For example, although the empirical models are simple, they may have an accuracy problem. In contrast, the field-solver-based techniques require huge amount of computation time even if they may yield the more accurate values than the empirical expressions. Further, both methodologies cannot fully reflect the realistic situations such as process variations and non-ideal characteristics of

the devices. Such deviations from the ideality become more serious with the advent of the deep-sub-micron process technologies. Thus, the experimental verification of them is essential for guaranteeing the signal integrity of the modern high-performance VLSI circuits.

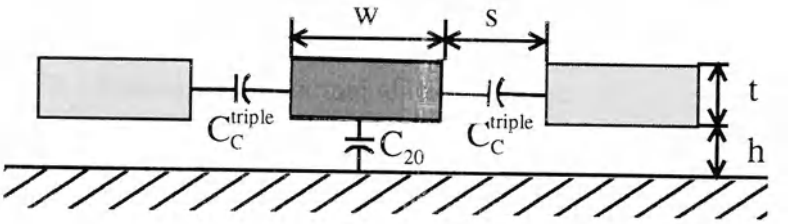


(a)



$$C_2 = C_{10} + C_C^{two}$$

(b)



$$C_3 = C_{20} + 2C_C^{triple}$$

(c)

Figure 1. Self capacitance and (a) simple line (b) two coupled lines (c) triple coupled lines.

Here, the field-solver-based empirical capacitance models which are accurate within about 10% error are introduced [2]. The models presented the capacitances in terms of the aspect ratio of the interconnect lines. The single line self-capacitance of Figure. 1(a) is

$$C_1 = \varepsilon_{ox}\left[2.80\left(\frac{t}{h}\right)^{0.222} + 1.15\left(\frac{w}{h}\right)\right]. \tag{5}$$

The self-capacitance for the two-coupled lines of Figure. 1(b) is

$$C_2 = C_1 + \varepsilon_{ox}\left[0.83\left(\frac{t}{h}\right) - 0.07\left(\frac{t}{h}\right)^{0.222} + 0.03\left(\frac{w}{h}\right)\right]\left(\frac{s}{h}\right)^{-1.34} \tag{6}$$

The self-capacitance for the triple-coupled lines of Figure. 1(c) is

$$C_3 = C_1 + 2\varepsilon_{ox}\left[0.83\left(\frac{t}{h}\right) - 0.07\left(\frac{t}{h}\right)^{0.222} + 0.03\left(\frac{w}{h}\right)\right]\left(\frac{s}{h}\right)^{-1.34} \tag{7}$$

The coupling capacitance for two coupled lines of Figure. 1(b) is

$$C_c^{two} = \varepsilon_{ox}\left[1.82\left(\frac{t}{h}\right)^{1.08} + \left(\frac{w}{h}\right)^{0.32}\right]\left(\frac{s}{h} + 0.43\right)^{-1.38} \tag{8}$$

The coupling capacitances for triple coupled lines of Figure. 1(c)is

$$C_c^{triple} = \varepsilon_{ox}\left[1.93\left(\frac{t}{h}\right)^{1.1} + 1.14\left(\frac{w}{h}\right)^{0.31}\right]\left(\frac{s}{h} + 0.51\right)^{-1.45} \tag{9}$$

Note, even if the above equations for the first level interconnects have been widely used for CAD timing verification, these cannot be directly applied for the upper layer line capacitances. For the multi-layer interconnect capacitance determination, 3-dimensional characteristics should be included. There are some empirical models concerned with 3-D capacitances [33]-[36]. However, the 3-D capacitance determination is exceedingly difficult as well as time-consuming. Thus, industry, in practice, constructs their own libraries for the special structures and then modify them for their practical applications.

2.1.2 Low Fequency Capacitance Measurement Techniques

While the analysis and synthesis of overall system characteristics are possible through theoretical approaches, practical problems including non-ideal or unexpected effects are found through experiments. Experiments and theory are indispensable to the circuit designer to characterize an unknown system. Particularly, as long as it is concerned wth the capacitances, experimental verifications are essential since there may be, in practice, large deviation from simulation results. That is, although both the capacitance and resistance plays a pivotal role in interconnect circuit performance, knowing the experimental capacitance values is vital to determine the circuit delay and crosstalk noise (coupling noise) on an IC chip.

In low frequency characterization using open and short circuit measurement techniques, capacitance measurements can be easily achieved. Capacitance is calculated from the relationship between a short circuit measurement and the definition of capacitance [37]. However, since short and open circuit cannot be easily implemented at high frequencies, other measurement techniques must be used for the high frequency capacitance characterization.

There are several capacitance definitions such as capacitive induction coefficients, conventional SPICE-like capacitance, and short circuit capacitance, although they are related with one another. Among them the only measurable definition of capacitance is the short circuit capacitance (particularly in multi-conductor systems) even if the others are conceptually physical and easy to understand. In this section, conventional capacitance and short circuit capacitance are related to each other. Then, a new concept of two terminal capacitance which is measurable quantity is presented. Finally, short circuit capacitance is represented in terms of the two terminal capacitance.

Now let us start with the introduction of the self capacitance and mutual capacitance. In n-conductor systems the charge per unit length on the i-th conductor, Q_i, is given by

$$Q_i = C_{ii}V_i + \sum_{j=1, j\neq i}^{n} C_{ij}\,(V_i - V_j) \tag{10}$$

where

$V_i \equiv$ voltage of the i-th conductor with reference to ground

$C_{ii} \equiv$ capacitance between i-th conductor and ground

$C_{ij} \equiv$ coupling capacitance between i-th conductor and j-th conductor.

Note, the above C_{ii} and C_{ij} are definitions for a SPICE-like simulation program. However, for the theoretical analysis, (10) can be rewritten in terms of a short circuit capacitance matrix. The short circuit capacitance matrix which is analogous to the node admittance matrix is defined as [38],

$$[C_s] \equiv \begin{bmatrix} C_{s11} & \cdots & C_{s1n} \\ \vdots & \ddots & \vdots \\ C_{sn1} & \cdots & C_{snn} \end{bmatrix} \tag{11}$$

where

$$C_{sji} = C_{sii} \equiv \sum_{j=1}^{n} C_{ij} \qquad \text{for } j = i \tag{12}$$

$$C_{sij} \equiv -C_{ij} \qquad \text{for } j \neq i. \tag{13}$$

Thus (10) can be rewritten as,

$$Q_i = \sum_{j=1}^{n} C_{sij} V_j \tag{14}$$

where C_{sij} is the short circuit capacitance. The main reason why a capacitance is defined as the short circuit capacitance matrix is that it is easier to obtain short circuit capacitances either from the experimental measurements or from the numerical computations. The relationship between the conventional capacitance definition, C_{ij} and the short circuit capacitance definition, C_{sij} of (12) is,

$$C_{sij} = -C_{ij} \quad \text{for} j \neq i. \tag{15}$$

$$C_{sii} = \sum_{j=1}^{n} C_{ij} \tag{16}$$

that is,

$$C_{ii} = \sum_{j=1}^{n} C_{sij} \tag{17}$$

Now, defining "A" as an active node set, if nodes are assigned to active node set or inactive node set by following criteria [37]

$$V_i = V \quad if \ \ i \in A \tag{18}$$

$$V_i = 0 \quad if \ \ i \notin A, \tag{19}$$

the two terminal capacitance per unit length (easily measured) is defined as the capacitance between these two subsets. In other words, the two terminal capacitance (C_{2T}^{A}) between all node set $x \in A$ (active node set) and all node set $y \notin A$ (inactive node set) is

$$C_{2T}^{A} = \sum_{i \in A} \frac{Q_i}{V}. \tag{20}$$

Then, the short circuit capacitance can be represented in terms of the two terminal capacitance,

$$C_{sii} = C_{2T}^{i} \tag{21}$$

$$C_{sij,i \neq j} = \frac{1}{2}\left(C_{2T}^{\{i,j\}} - C_{2T}^{\{i\}} - C_{2T}^{\{j\}}\right) \tag{22}$$

Thus, n-coupled conductor capacitances can be readily determined by performing the repeated 2-port network measurements.

2.1.3 Interconnect Inductance (L)

Although the previous RC models are very useful for a simple first order approximation technique, they may not be accurate enough for modern high-speed VLSI circuit timing verification since they neglected proximity effects, skin effects, and silicon substrate effects. These significantly affect the IC interconnect responses at high frequencies (e.g., propagation delay, rise time, and fall time). Thus, the RC models are not suitable for such circuits that require signal transmission bandwidths over 1GHz. In order to maintain a sharp controlled edge on a clock it will be necessary to have a good model of the clock components up to $5 \sim 10$ times the clock frequency [39]. Thus, the RC model will not accurately predict clock edge on the IC in the 100MHz to 200MHz regions and beyond. A rule of thumb relates the bandwidth to a rise time [40]

$$BW \approx \frac{0.35}{t_{rise}}. \tag{23}$$

Thus, in order to accurately model the propagation of 100ps transmissions on the IC, the interconnect models must be accurate to 3.5 GHz. This is clearly

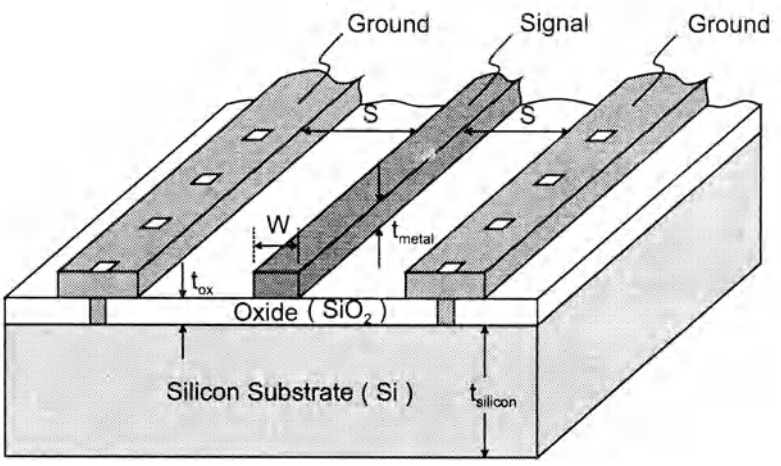


Figure 2. A typical coplannar interconnect structure.

not the case for the RC model. Thus, in the near future, microwave characterization techniques and complicated distributed circuit models which include the interconnect inductance may become necessary for high-speed VLSI circuits.

Unlike the electric field, since the magnetic field penetrates the silicon substrate, the silicon substrate effect should be taken into account to calculate inductance matrix. The inductance matrix for lossless (or low lossy) lines is commonly determined by using [41]

$$[L] = \mu_o \varepsilon_o [C]^{-1} . \tag{24}$$

However, (24) is not accurate enough to be employed for lossy lines. Further, (24) did not consider the silicon substrate effect. Thus, the inductance matrix is determined by introducing an effective dielectric constant (a kind of fitting parameter) which can take both the silicon substrate effect and lossy effect into account. The effective dielectric constant is determined from a single transmission line as shown in Figure. 2 and then the value is applied for the multi-line inductance matrix determination. Shibata and Sano [22] numerically estimated the magnetic power along the silicon substrate in the coplanar MIS (metal-insulator-semicondutor) structure and showed that most of magnetic energy is confined within a depth "d" from the silicon surface which is given by

$$d = 2s + w \tag{25}$$

where "s" is the spacing between the signal line and ground line in a coplanar structure and "w" is a signal line width (see Figure. 2). Thus, introducing the result, an empirical inductance formulae of a silicon-based IC interconnect can be used to determine the inductance [16]. It is given by

$$L_s = \mu_o \frac{1}{2\pi} ln \left[\left(\frac{d}{0.59w} + 1.1 \right) - 0.5 + \sqrt{\left(\frac{d}{0.59w} + 1.1 \right)^2 - 1.05} \right]. \tag{26}$$

The model (26) shows good agreement with s-parameter-based inductance measurements within 10% error. The effective dielectric constant, ε_{reff}, can be easily calculated with a single line capacitance (C_s) and (26) as follows

$$\varepsilon_{reff} = \frac{C_s L_s}{\mu_o \varepsilon_o}. \tag{27}$$

Since the magnetic field penetration depth of the multi-line system is approximately equal to that of the single line, the effective dielectric constant can be used for the multi-line system. Thus, the inductance matrix for the multiple lines can be yielded by

$$[L] \approx \mu_o \varepsilon_o \varepsilon_{reff} [C]^{-1}. \tag{28}$$

Note, the inductances which consider the silicon substrate effect show much larger values than those without the effect.

2.2 Interconnect Transmission Line Circuit Model

For the purpose of the timing verification of IC interconnects, ideal ground-based RC- or RLC-distributed-circuit models for interconnect lines have been widely employed up to date [3][6][9][14]. However, such models for the IC interconnects fabricated on a conductive silicon substrate have inherent limitations without considering the silicon substrate effect and frequency-variant transmission line characteristics [9]-[12] [19]-[24][26][27][42]-[46]. However, their characteristic variations due to the complicated silicon substrate and the operating frequency make it inherently difficult to understand and characterize the integral aspect of signal transients on such interconnect lines [19][24][26]. In fact, the difficulties result from the intrinsic attributes of the transmission line system. That is, the propagation mode of the frequency components of a pulsed signal on an IC interconnect may changes

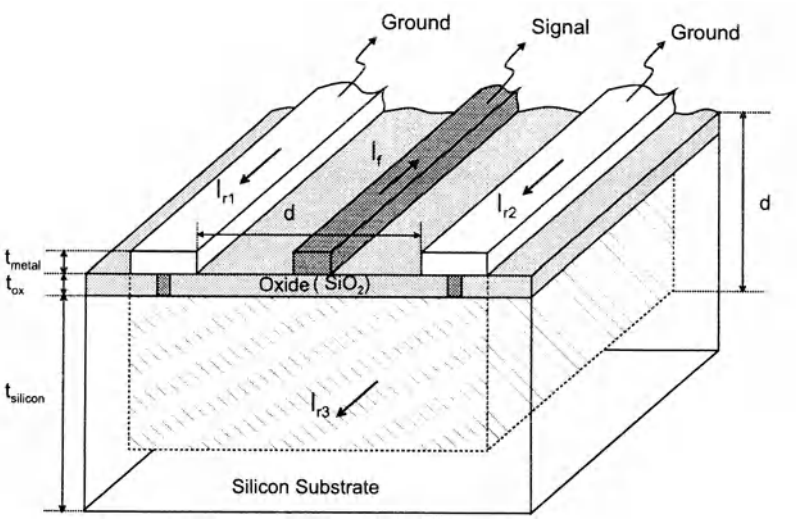


Figure 3. A combined transmission line system composed of a co-planar structure and a microstrip. There are three current return paths. The dotted region means the effective ground plane of the silicon substrate.

with the operation frequency and substrate resistivity, thereby making the system behavior sophisticated [11], [26], [44]. As another attribute of the system, the parametric variations due to the silicon substrate and current return path impedance (i.e., the proximity effects) of the combined system of the co-planar lines and micro-strip line as in IC interconnects make the system behavior much more complicated. In spite of such difficulties, an IC interconnect transmission line model considering all these effects becomes essential for the high-performance VLSI circuit design [11][20][26][44].

2.2.1 Physical Characteristics of Interconnect Transmission Line

An IC interconnect is a combined system composed of a coplanar structure and a micro-strip [26]. In order words, the IC interconnect structure has several ground return paths as shown in Figure. 3. That is, a conductive silicon substrate acts as a kind of an imperfect ground return path. The IC interconnect transmission line parameters are inherently frequency-variant because of the silicon substrate effect, the metal skin effect, and the proximity effect in the current return paths. The signal on the interconnect line of the MIS (metal-insulator-silicon) structure propagates with one of the three propagation modes due to a silicon substrate resistivity and an

operating frequency [19]. Since the conductive silicon substrate does not act as a perfect ground plane, significant amount of return currents comes back through the ground metals which are on the same metal layer as the signal line. Furthermore, over the frequency range of interest, because of the skin effect and proximity effect, the changes in resistances and inductances can be substantial [11]. However, although the system is intrinsically very intricate, its physical phenomena can be fairly simplified if the minor effects of the system are neglected.

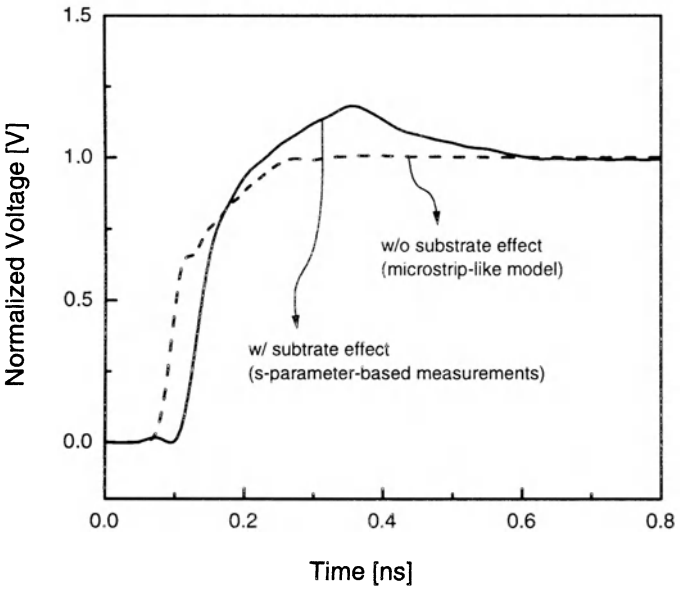


Figure 4. Unit step responses with the silicon substrate effect and without the silicon substrate effect. The line width is 2 μm and line length is 8 mm.

The silicon substrate resistivity of today's IC is not so low as to be the skin effect mode. Further, since it is not so high as to be the dielectric quasi-TEM mode, digital signal propagates with the slow wave mode rather than with the dielectric quasi-TEM mode. Therefore, as modeled by Hasegawa [19], in the slow wave mode, the transmission line effects of the signal line can be modeled with an RLC circuit. In the mode, the signal line capacitance is nearly equal to the oxide capacitance. In contrast, the inductance cannot be simply modeled as in capacitance. The return currents come back through

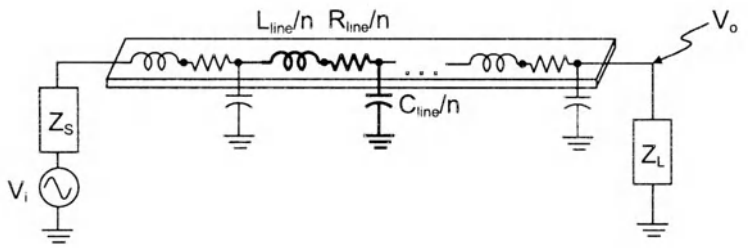


Figure 5. The ideal transmission line model of IC interconnect line with n cascaded-lumped RLC components.

several current return paths and the considerable amount of the currents is returned through the co-planar ground lines. That is, most of the magnetic energy is confined within a vertical distance depending on the geometrical structure of the system ground path [22]. Thus, all these ground paths must be taken into account for an accurate circuit model development. In order to more clearly show the substrate effect, the unit-step responses of the structure of Figure. 3 with the silicon substrate effect and without the silicon substrate effect (i.e., the microstrip-like model) are shown in Figure. 4, where the line width is 2 μm and line length is 8 mm. The differences between the two are apparent.

2.2.2 Circuit Model with Proximity Effect and Skin Effect

A transmission line with an ideal system ground can be fairly well modeled with the lumped RLC ladder network as shown in Figure. 5. That is, if the length of a segmented line is much shorter than the wavelength of the highest frequency component (e.g., about 10 times smaller than the wavelength of the clock frequency), the distributed phenomena can be accurately modeled with the segmented ladder network. For most of the IC interconnect lines, the transmission line effects for a low frequency operation can be well described with the circuit model of Figure. 5 because the return path impedances are negligibly small. However, the effective transmission line parameters are changed with the increase of the operation frequency due to the skin effect and proximity effect. Thus, the proximity effect and skin effect must be included in order to accurately model the more practical physical phenomena of the transmission line effects. In order to reflect them, the transmission line of the IC interconnect must be modified as shown in

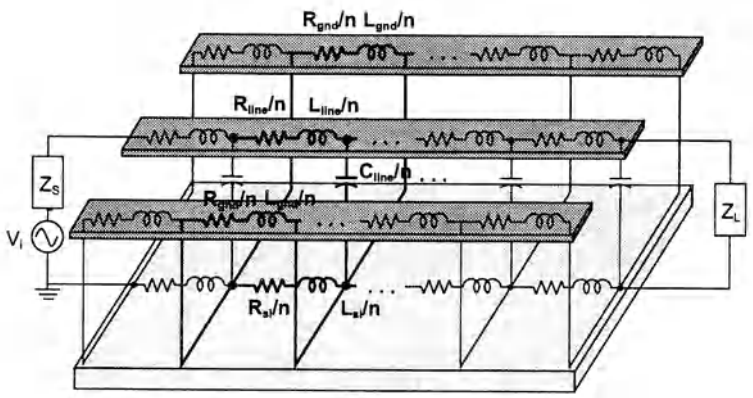


Figure 6. The transmission line model with the current return path for the proximity effect.

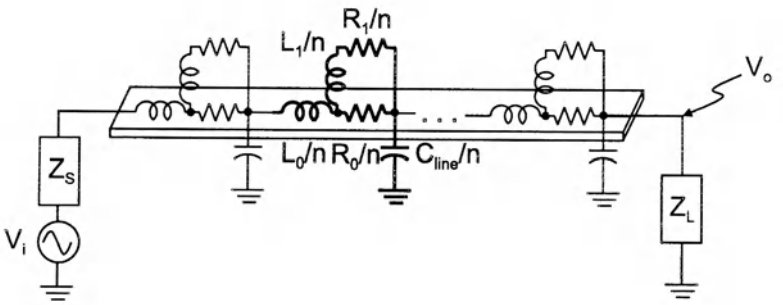


Figure 7. The modified transmission line model with the constant model parameters including both the proximity effect and the skin effect.

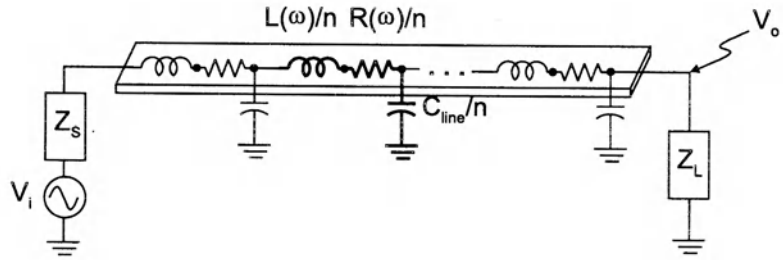


Figure 8. The modified transmission line model with the frequency-variant model parameters including both the skin effect and the proximity effect.

Figure. 6 and Figure. 7, respectively. Note that the transmission line parameters of Figure. 7 should be determined by solving the combined circuit equations for the circuits of Figure. 6 and Figure. 7. The circuit equations can be established by assuming that the return paths of currents are dominated with the resistances at the low frequency while they are dominated with the inductive reactances at the high frequency. Then the transmission line parameters of Figure. 7 become [11]

$$R_0 = \frac{R_{dc}L_{dc}}{L_{hf}} \quad \text{and} \quad R_1 = \frac{R_{dc}L_{dc}}{L_{dc} - L_{hf}}, \tag{29}$$

$$L_0 = L_{hf} \quad \text{and} \quad L_1 = \frac{L_{dc}^2}{L_{dc} - L_{hf}}, \tag{30}$$

where R_{dc}, L_{dc}, and L_{hf} are low frequency resistance, low frequency inductance, and high frequency inductance, respectively. The low frequency and high frequency components are given by

$$R_{dc} = R_{line} + R_{GND} = R_{line} + \frac{R_{gnd}R_{si}}{R_{gnd} + 2R_{si}} \tag{31}$$

$$L_{dc} = L_{line} + 2\left(\frac{R_{GND}}{R_{gnd}}\right)^2 L_{gnd} + \left(\frac{R_{GND}}{R_{si}}\right)^2 L_{si} \tag{32}$$

$$L_{hf} = L_{line} + \frac{L_{si}L_{gnd}}{L_{gnd} + 2L_{si}}. \tag{33}$$

Note that $R_{GND} \equiv R_{g1}//R_{g2}//R_{g3}$, where R_{gi} means the i-th ground path resistance [47]. Thus, if the circuit model of Figure. 7 is changed into that of Figure. 8, the circuit model parameters of Figure. 8 which include the proximity effect and skin effect can be yielded as

$$R(\omega) = \frac{R_0R_1(R_0 + R_1) + \omega^2R_0L_1^2}{(R_0 + R_1)^2 + \omega^2L_1^2} \tag{34}$$

$$L(\omega) = L_0 + \frac{L_1R_0^2}{(R_0 + R_1)^2 + \omega^2L_1^2}. \tag{35}$$

Note that, in the modified circuit model of Figure. 8, the transmission line parameters differ from the conventional circuit model parameters of Figure. 5, although the superficial circuit topology seems to be identical. The differences between the two models will be discussed in more detail in the ensuing section concerned with the signal delay model.

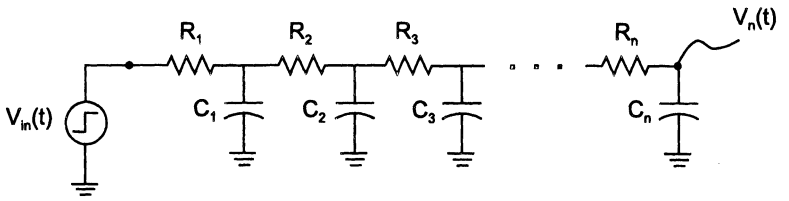


Figure 9. RC circuit model of IC interconnect.

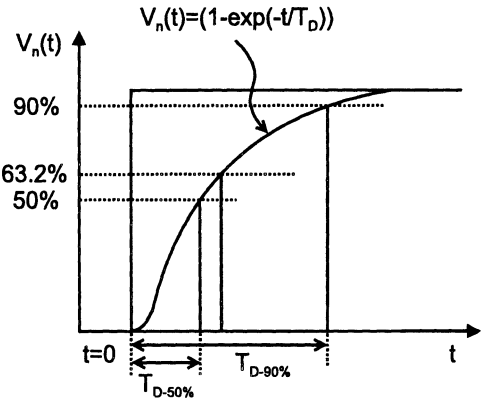


Figure 10. Schematic description of Elmore delay.

3 Signal Delay Modeling due to Interconnects

3.1 Analytical Delay Models in RC Network

The RC time delay for the circuit of Figure. 9 can be readily estimated with the Elmore delay model which is widely used as a first order interconnect timing verification model[4]. Although the Elmore delay model is not accurate for the modern high-speed integrated circuits, it is the simplest delay model which is very useful for a simple hand-analysis. The step response of the circuit is schematically shown in Figure. 10 and is given by [49]

$$V_n(t) = \left(1 - exp\left(-\frac{t}{T_D}\right)\right) \tag{36}$$

where T_D is the time that $V_n(t)$ arrives at 63.2% of the final value and is given by

$$T_D = \sum_{i=1}^{n} R_i \sum_{j=i}^{n} C_j \tag{37}$$

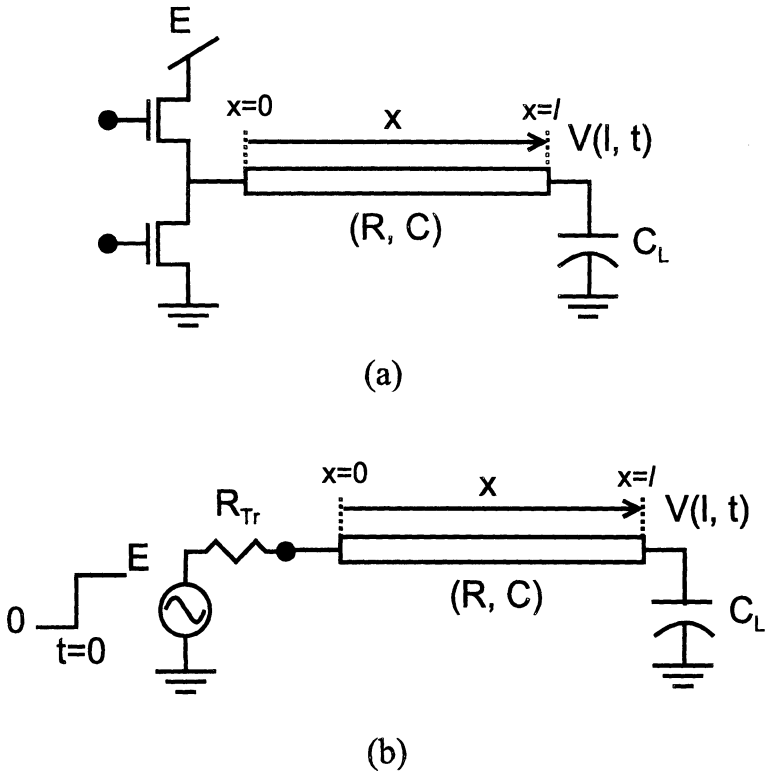


Figure 11. An interconnection circuit model (a) driving circuit with capacitive load and (b) RC circuit model.

Therefore, the 50% delay and 90% delay can be readily determined by [4][5]

$$T_{D-50\%} = (ln2) \sum_{i=1}^{n} R_i \sum_{j=i}^{n} C_j \tag{38}$$

$$T_{D-90\%} = (ln10) \sum_{i=1}^{n} R_i \sum_{j=i}^{n} C_j, \tag{39}$$

respectively. These two equations are called Elmore delay model.

Although the Elmore delay model is simple, it is based on a single dominant pole approximation. Thus, since it is not accurate enough for the long transmission line delay estimation, there have been many efforts to improve the interconnect line delay model [49][3][5]. Among them, Sakurai developed an accurate improved delay model for the RC network. The normalized voltage at the end of the line as shown in Figure. 11 can be expressed in a series

expanded form [8]

$$V(l,t) = 1 + \sum_{k=1}^{\infty} K_k e^{-\sigma_k t/RC} \approx 1 + K_1 e^{-\sigma_1 t/RC} \tag{40}$$

where

$$K_1 = -1.01 \frac{(R_{Tr}/R) + (C_L/C) + 1}{(R_{Tr}/R) + (C_L/C) + \pi/4} \tag{41}$$

and

$$\sigma_1 = \frac{1.04}{(R_{Tr}/R)(C_L/C) + (R_{Tr}/R) + (C_L/C) + (2/\pi)^2}. \tag{42}$$

Thus, the 50% delay and 90% delay can be readily determined from (40) by

$$T_{D-50\%} = 1.02RC + 2.3(R_{Tr}C_L + R_{Tr}C + RC_L) \tag{43}$$

$$T_{D-90\%} = 0.377RC + 0.693(R_{Tr}C_L + R_{Tr}C + RC_L) \tag{44}$$

These simple delay models are in general more accurate than those of the Elmore.

Although the aforementioned one pole approximation techniques are simple as well as efficient, it has inherent problems since the interconnect circuits have many poles and zeros. Unlike the previous one-pole-dominant approximation techniques, multi-pole approximation technique is recently developed which is named as AWE (Asymptotic Waveform Approximation or model order reduction technique) [14]. The AWE technique is much more accurate than the previous one pole model since it takes multiple poles and zeros of a system into account. The AWE technique determines the poles and zeros by using the approximated rational function which is fitted with the original function. Thus, if the origional function varies monotonically as in RC-time-constant dominant circuits, it is very stable as well as accurate. However, if the L/R time constant of a system dominates the RC-time constant of the system, the original function cannot be accurately approximated with the fitted rational function. In such cases, the AWE technique may have a significant stability problem. Although the stability problems have been improved, it is an inherent problem of the technique.

3.2 Analytical Delay Models in RLC Network

Since the RC delay models are not accurate enough to estimate today's high-speed VLSI circuit timing verification, novel delay models based on more accurate RLC network have been developed [9][47]. However, unlike the RC network, the RLC network have an inherent difficulty in the formation of a simple closed form model since the RLC models have complicated imaginary terms. Recently Khang developed a closed form delay model for the RLC network, assuming two dominant poles of the circuits[9]. However, since it requires many empirical fitting parameters, it is not a complete analytical model. Further, the model does not take the skin effect, substrate effect, and proximity effect into account. An improved RLC circuit delay model was developed by Jin et al.[47]. The model is based on the accurate RLC network which is described in the previous section. For the analytical delay estimation due to a transmission line, an interconnect system function is approximated with two poles [9]

$$H(s) \approx \frac{1}{1 + b_1 s + b_2 s^2} \tag{45}$$

with frequency-independent coefficients, b_1 and b_2. The "s" is the Laplace transform variable. Note that, the coefficients(i.e., b_1 and b_2) should be frequency-variant if the proximity effect and skin effect are taken into account. The frequency-variant coefficients of (45) can be yielded by using the circuit model of Figure. 8[47],

$$b_1(\omega) = R_S(C_{line} + C_L) + \left(\frac{C_{line}}{2} + C_L\right) R_p \tag{46}$$

where

$$R_p \equiv \left[\frac{R_0 R_1 (R_0 + R_1) + \omega^2 R_0 L_1^2}{(R_0 + R_1)^2 + \omega^2 L_1^2}\right]$$

and

$$b_2(\omega) = \frac{R_S C_{line}}{2}\left(\frac{C_{line}}{3} + C_L\right) R_p + \frac{C_{line}}{6}\left(\frac{C_{line}}{4} + C_L\right) R_p^2$$
$$+ L_S(C_{line} + C_L) + \left(\frac{C_{line}}{2} + C_L\right)\left[L_0 + \frac{L_1 R_0^2}{(R_0 + R_1)^2 + \omega^2 L_1^2}\right] \tag{47}$$

where R_S and L_S are the source resistance and source inductance (i.e., $Z_s = R_s + j\omega L_s$) (see the model of Figure. 6 and Figure. 7 for parameter

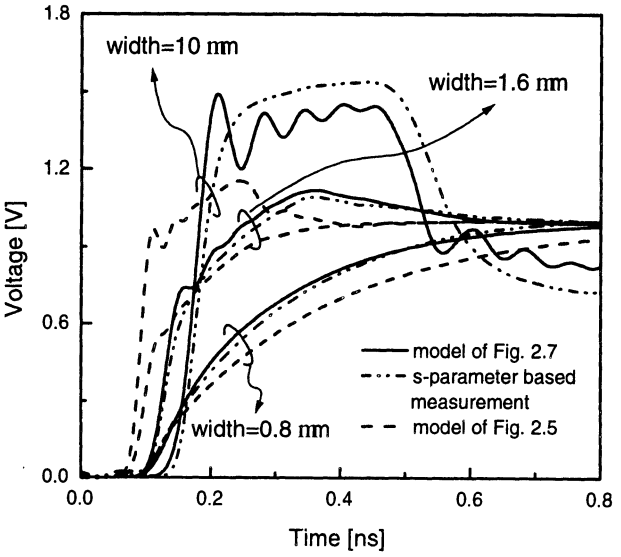


Figure 12. Signal transient responses for various test patterns.

definitions). Although the coefficients are frequency-variant, the delay is mainly concerned with low frequency components while the high-frequency components are mainly concerned with the spikes of the waveshape. Thus, with $b_1(\omega = 0)$ and $b_2(\omega = 0)$, the 50%- and 90%-delay can be approximately calculated by using

$$b_1 \approx R_S(C + C_L) + \left(\frac{C}{2} + C_L\right)\left[\frac{R_0 R_1 (R_0 + R_1)}{(R_0 + R_1)^2}\right] \tag{48}$$

$$\begin{aligned} b_2 \approx\ & \frac{R_S C}{2}\left(\frac{C}{3} + C_L\right)\left[\frac{R_0 R_1 (R_0 + R_1)}{(R_0 + R_1)^2}\right] \\ & + \frac{C}{6}\left(\frac{C}{4} + C_L\right)\left[\frac{R_0 R_1 (R_0 + R_1)}{(R_0 + R_1)^2}\right]^2 \\ & + L_S(C + C_L) + \left(\frac{C}{2} + C_L\right)\left[L_0 + \frac{L_1 R_0^2}{(R_0 + R_1)^2}\right] \end{aligned} \tag{49}$$

Then the delay becomes

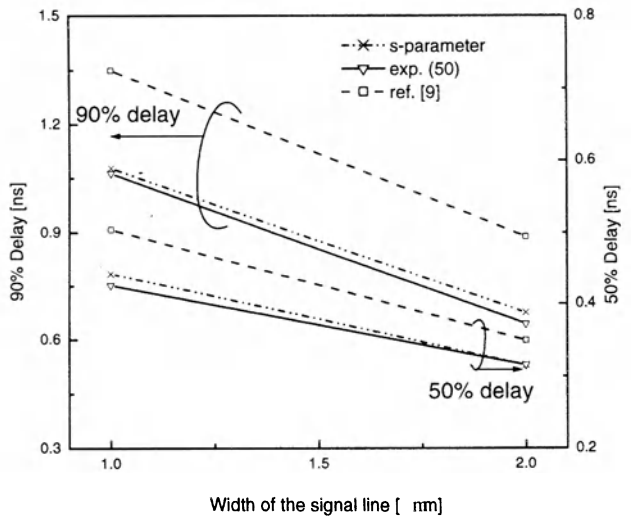


Figure 13. Signal delay for various techniques.

$$T_{delay} = \begin{cases} K_r \frac{2b_2}{b_1 - \sqrt{b_1^2 - 4b_2}} & \text{for} \quad b_1^2 - 4b_2 > 0 \\ K_c \frac{2b_2}{\sqrt{4b_2 - b_1^2}} & \text{for} \quad b_1^2 - 4b_2 < 0 \\ K_d \frac{b_1}{2} & \text{for} \quad b_1^2 - 4b_2 = 0 \end{cases} \tag{50}$$

where K_r, K_c and K_d are given by

$$K_r = \ln\left(\frac{1}{2\,(1 - V_{th})}\left[1 + \frac{b_1}{\sqrt{b_1^2 - 4b_2}}\right]\right) \tag{51}$$

and

$$K_c = \frac{(1 - V_{th})\, e^{\frac{b_1}{2b_2} T_{Elmore}}}{\sqrt{1 + \left(\frac{b_1}{\sqrt{4b_2 - b_1^2}}\right)^2}} - \tan^{-1}\left(\frac{\sqrt{4b_2 - b_1^2}}{b_1}\right) \tag{52}$$

$$K_d = 3.9. \tag{53}$$

The 50%- and 90%-delay for the model and measurement data are compared in Figure. 12 and Figure. 13, respectively. As shown in Figure. 12 and Figure. 13, the signal delays without considering the skin effect, proximity effect, and silicon substrate effect show large differences with those of the s-parameter-measurement-based values. Particularly, for the wider lines, the effects are much more evident. Thus, the model without the effects cannot be accepted for today's high-performance VLSI circuit timing verification any more. In contrast, the signal delays using the modified transmission line model (i.e., the model of Figure. 8 which includes the skin-effect, proximity effect, and the silicon substrate effect show much more excellent agreements with the s-parameter-measurement-based signal transient data which will be discussed in the next section in more detail.

3.3 S-Parameter-Based Signal Transient Characterization

As the circuit speed dramatically increases, the today's VLSI circuit design necessarily requires experimental verification of the models. The s-parameter measurement technique is extremely stable as well as accurate up to a very high-frequency. Moreover, it is very easy as well as accurate to de-embed the pad parasitic for on-wafer characterizations. In this section, time-domain signal-transient characterization technique of an IC interconnect with s-parameters is introduced[80]. The transmission line characteristics of an IC interconnect line can be mathematically formulated with the Telegrapher's equations which are differential equations. The solutions of the equations in terms of voltage and current in the frequency domain are

$$V(x,\omega) = V_A e^{-\gamma(\omega)x} + V_B e^{\gamma(\omega)x} \tag{54}$$

and

$$I(x,\omega) = \frac{1}{Z_C(\omega)} \left(V_A e^{-\gamma(\omega)x} - V_B e^{\gamma(\omega)x} \right), \tag{55}$$

where $\gamma(\omega)$ is the propagation constant and $Z_C(\omega)$ is the characteristic impedance. The coefficients V_A, V_B, I_A, and I_B can be determined with boundary conditions. The $\gamma(\omega)$ and $Z_C(\omega)$ can be represented with the PUL(per-unit-length)-transmission line parameters as

$$\gamma(\omega) = \sqrt{(R + j\omega L)(G + j\omega C)}, \tag{56}$$

and

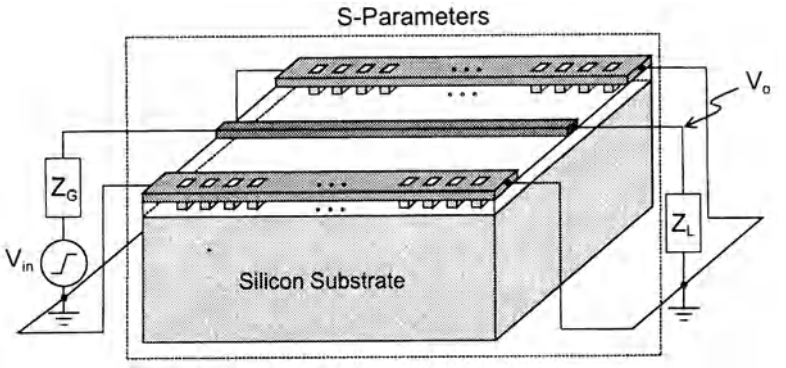


Figure 14. Interconnect system in terms of s-parameters.

$$Z_C(\omega) = \sqrt{(R + j\omega L)/(G + j\omega C)}, \tag{57}$$

respectively. Hence, the accurate transmission line parameter determination is substantially important. However, regardless of process variations and non-ideal characteristics of them, it is very difficult to accurately determine them since they are all functions of frequency. The difficulties in the characterization of the IC interconnect on SiO_2-Si substrate are primarily concerned with the complicated silicon substrate effect and frequency-variant transmission line parameters. Instead of using the frequency-variant RLCG parameters, an interconnect system is newly defined in terms of measured s-parameters as shown in Figure. 14. Since the s-parameters can comprise of all the parametric variations concerned with the IC interconnect in a broad frequency band, aforementioned problems can be overcome, thereby investigating the accurate system characteristics. The interconnect system function can be mathematically formulated from (54) and (55) with boundary conditions in the frequency domain as

$$H(\omega) = \frac{Z_C(\omega)}{Z_C(\omega) + Z_G(\omega)} \frac{exp(-\gamma(\omega) l) + \rho_L(\omega) exp(-\gamma(\omega) l)}{1 - \rho_G(\omega) \rho_L(\omega) exp(-2\gamma(\omega) l)}, \tag{58}$$

where the reflection coefficients are defined as

$$\rho_G(\omega) \equiv \frac{Z_G(\omega) - Z_C(\omega)}{Z_G(\omega) + Z_C(\omega)} \tag{59}$$

and

$$\rho_L(\omega) \equiv \frac{Z_L(\omega) - Z_C(\omega)}{Z_L(\omega) + Z_C(\omega)}. \tag{60}$$

The impedances $Z_G(\omega)$ and $Z_L(\omega)$ are denoted as the input source impedance and output load impedance, respectively. The parameter l in (58) denotes the line length. Thus, once $H(\omega)$ is determined, the frequency-domain response $V_o(\omega)$ of the interconnect line can be simply formulated with

$$V_o(\omega) = V_i(\omega) H(\omega), \tag{61}$$

where $V_i(\omega)$ is the input signal. The transmission line parameters are given in terms of s-parameters by [24]

$$e^{-\gamma(\omega)l} = \left(\frac{1 - S_{11}^2 + S_{21}^2 \pm \sqrt{\left(S_{11}^2 - S_{21}^2 + 1\right)^2 - (2S_{11})^2}}{2S_{21}} \right)^{-1} \tag{62}$$

and

$$Z_C^2(\omega) = Z_o^2 \frac{(1 + S_{11})^2 - S_{21}^2}{(1 - S_{11})^2 - S_{21}^2}, \tag{63}$$

where Z_o denotes the reference impedance of the s-parameter measurement system. Thus (58) can be characterized in the frequency domain once the s-parameters are acquired. Note that all the s-parameters are functions of frequency. That is, $S_{ij} = S_{ij}(\omega)$. Thus, the measured s-parameters implicitly include all the frequency-variant transmission line characteristics. Since the s-parameter-based system function is of discrete value at each measurement point, the transfer function between the measured frequency points can be approximately interpolated as

$$H(\omega_i \le \omega \le \omega_j) \approx \frac{H(\omega_j) - H(\omega_i)}{\omega_j - \omega_i}\omega + B \quad \text{for} \quad \omega_i \le \omega_j, \tag{64}$$

where ω_i and ω_j are the i-th and j-th measurement frequency points. The constant B in (64) can be determined as

$$B = H(\omega_j) - \frac{H(\omega_j) - H(\omega_i)}{\omega_j - \omega_i}\omega_j. \tag{65}$$

In addition, since the s-parameter data are acquired at finite frequency points (e.g., usually 200 measurement points) that are within a finite frequency band (usually 50MHz to 20GHz), the low frequency data below

50MHz and the high frequency data above 20GHz are extrapolated from the measured data. Finally, in order to completely characterize the frequency domain response of (61), the input function should also be formulated in the frequency domain. The input pulse of $V_i(\omega)$ with both rise time (t_r) and fall time (t_f) can be mathematically formulated in the frequency domain as n-delayed step pulses as [48]

$$V_i(\omega) = \sum_{k=1}^{n} \frac{A}{n} t_f sinc\left(\frac{\omega t_f}{2}\right) exp\left(-\frac{j\omega t_f}{2}\right) exp\left(-\frac{j\omega(k-1)t_r}{n}\right), \quad (66)$$

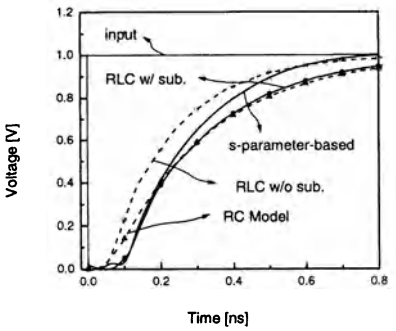
where A is the magnitude of the pulse. Thus, combining (62) to (66), the frequency domain response can be represented with the s-parameters. As a result, the time domain response $v_o(t)$ of (61) can be directly yielded by the inverse transform of (61) into the time domain. That is, without the transmission line parameter extraction step, the time-domain signal transients can be directly determined.

The s-parameter-measurement-based output responses for different line widths are compared with the constant RC- and RLC-measurement-based SPICE simulation in Figure. 15. The s-parameter-based signal responses show much faster rise time than those of the RC- or RLC-based data. Moreover, their waveshapes are considerably different from the s-parameter-based signal transients. In many critical paths such as in clock distribution networks, thick and wide lines are used to reduce line resistance. This is primarily because the resistance has a more dominant influence upon the signal delay than the capacitance. However, such a simple design scheme may result in a catastrophic design failure since the inductance effect of the low resistive line is crucial. The large ringing (i.e., oscillation) due to the inductance results in a large setting time delay. Obviously, the interconnect models without the experimental characterization-based verification may not be exact enough to meet the timing budget as well as noise margin concerned with today's high-performance VLSI circuits or next generation VLSI circuits.

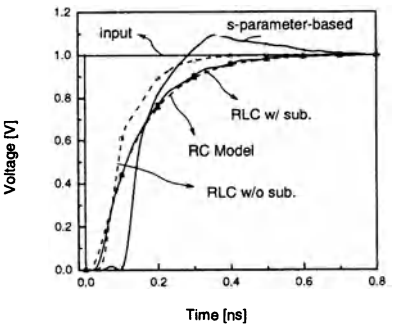
3.4 Further Consideration of Interconnect Delay Models

3.4.1 IC Interconnect Parameter Variations

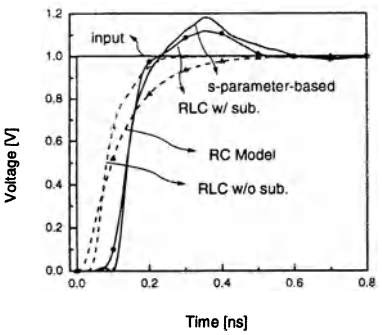
In today's silicon-based VLSI circuits, the signal propagation on interconnects on standard IC substrates exhibits a slow wave mode at low frequencies. However, it exhibits a quasi-TEM mode as frequency is increased



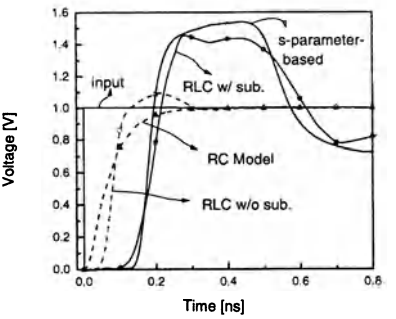
(a) line width=0.8 μm



(b) line width=1.6 μm



(c) line width=2.0 μm



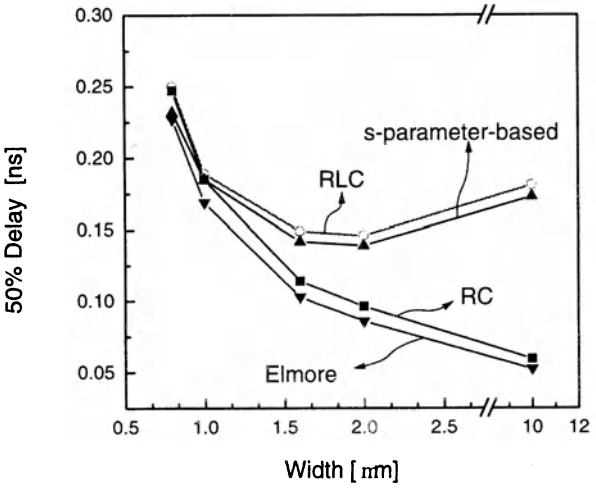
(d) line width=10.8 μm

Figure 15. Signal transient responses at 8mm long line for the step input signal. Note that "w/ sub" means "considering the silicon substrate effect" and "w/o sub" means "without considering the silicon substrate effect".

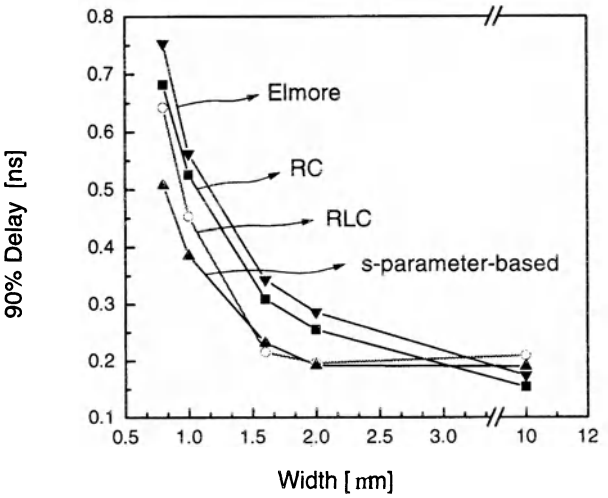
[19]. The strong interfacial polarization due to the mobile charges within the silicon substrate occurs in the slow wave mode. Due to the interfacial polarization, the electric field cannot penetrate the silicon substrate at low frequencies. That is, the mobile charges in the interface between the oxide layer and the silicon layer can screen out the electric field. In contrast, there exist strong magnetic flux linkages due to the interfacial charge variations within the silicon substrate. This physical phenomenon can be interpreted in terms of the magnetic field penetration. That is, the magnetic field can penetrate both the oxide layer and silicon substrate. Therefore, the interlinked magnetic flux due to interfacial polarization charge variations in the silicon substrate must be included for the inductance determination. In addition, the capacitance at low frequency becomes effectively reduced at high frequency since the propagation mode changes from the slow wave mode to the dielectric-quasi-TEM mode [19]. The resistance is increased as the frequency increases due to the skin effect [16][27]. The conductance is increased since the dielectric loss in the silicon substrate is increased as the frequency increases [16][27]. Consequently, all the transmission line parameters are fundamentally frequency-variant. However, since all the parametric variations of the transmission line with the frequency and silicon substrate characteristics are complicatedly entangled, the signal variations on the interconnect line cannot be simply described. The s-parameter-based signal transients can implicitly include all these parametric variations. Whereas the constant RC- or RLC-model do not accurately reflect these physical phenomena.

3.4.2 Signal Delays and Distortions due to Inductance

The RC delay for the circuit of Figure. 9 can be readily estimated with the Elmore delay model which is widely used as a first order interconnect timing verification model [4][5]. The signal delays for different line width are shown in Figure. 9 by varying the line width. The signal delay of RC model for the $10\mu m$ width line approximately shows the difference of 100ps using the s-parameter-based method. However, even for a large resistive line (i.e., no oscillation case), the RC-model does not accurately estimate the high-speed signal delay. The RLC-model of Figure. 8 more accurately estimate the delay than the RC-model. Nonetheless, the signal transient waveshapes of both RC- and RLC-model do not agree with those of the s-parameter-based signal transients. Even with the measured RLC values, is it not adequate enough to accurately estimate the signal integrity of the interconnect line.



(a) 50%delay



(b) 90%delay

Figure 16. Signal delay for the test patterns.

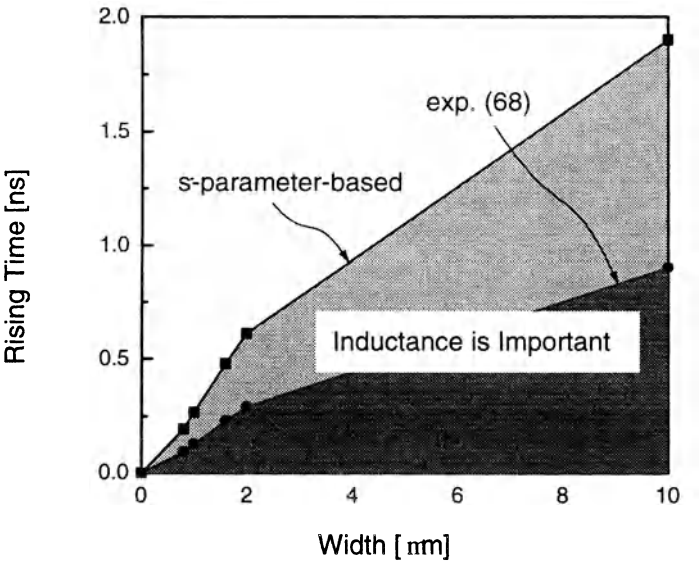


Figure 17. Region that the inductance effect is significant with the rise time and line width(the shaded regin).

Although the RLC-based signal transient characterization is much better than the RC-based one from the delay point of view, the RLC-based signal integrity verification may still lead to serious problems.

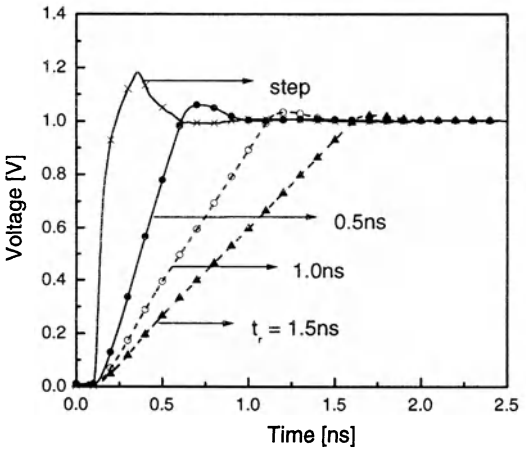
From the signal integrity point of view, the range of the line length in which inductance effects are significant in digital circuits can be approximately determined by combining transmission line characteristics with the lumped RLC circuit model as [12]

$$\frac{t_r}{2\sqrt{LC}} < l < \frac{2}{R}\sqrt{\frac{L}{C}}. \tag{67}$$

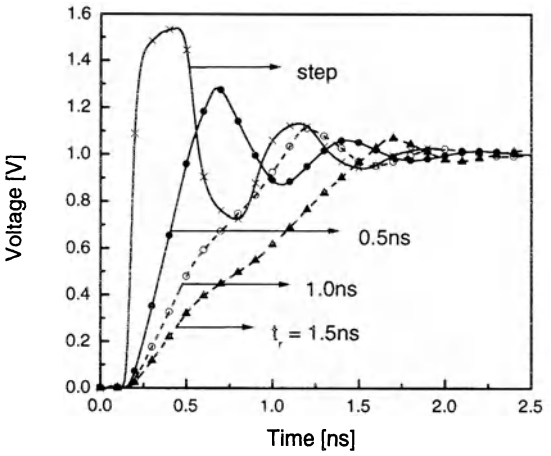
That is, if the rise time of a pulse is small enough to meet

$$t_r < 4\left(\frac{L}{R}\right) \tag{68}$$

the circuit shows oscillation, i.e., ringing phenomenon. For example, if a 0.8cm line with 10 μm width have about L=7nH and R=30Ω, t_r should be greater than 0.93nsec. Otherwise, the inductance may have a significant



(a) s-parameter-based signal transients for 2.0 μm with different rise times



(b) s-parameter-based signal transients for 10.0 μm with different rise times

Figure 18. Ringing phenomena with different rise times.

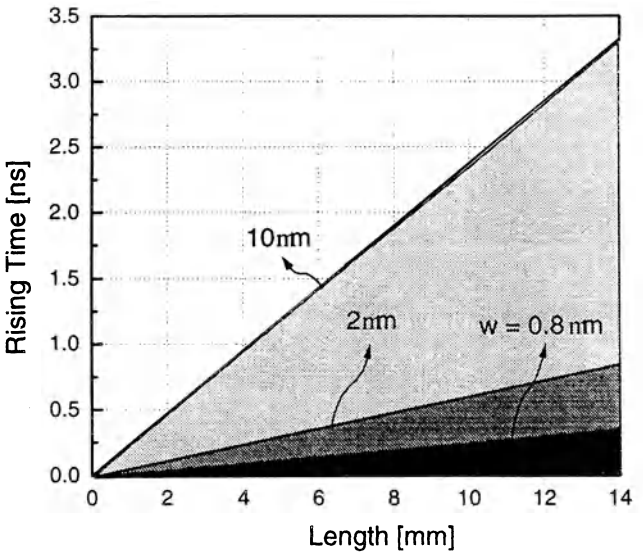


Figure 19. Region that the inductance effect is significant with rise time and line width. The inductance effect is significant below the lines.

effect on the circuit oscillation. Therefore, combining (26) with (68), the region in which the inductance effect is significant can be drawn in terms of the rise time and line width. However, it is noteworthy that (68) underestimates the effect by about 40% as shown in Figure. 3.10 and Figure. 3.11. In fact, this is because (67) is based on the lumped RLC circuit model. In the distributed transmission line circuits, RC time constant is about half of the lumped RC time constant. Thus (67) should be modified as

$$\frac{t_r}{2\sqrt{LC}} < l < \frac{3.36}{R}\sqrt{\frac{L}{C}}. \tag{69}$$

Accordingly, (67) becomes

$$t_r < 6.73\left(\frac{L}{R}\right). \tag{70}$$

This is very close to the s-parameter-based signal transient responses as shown in Figure. 17. Further, using (26) and (70), the region in which inductance effect is significant can be represented with a line length and a rise time as shown in Figure. 19. Clearly, the inductance effect is much more serious when both line resistance and switching time is small as shown in Figure. 19. This means that future interconnect lines such as a copper interconnect line and high aspect ratio (t/w) line may lead to a more serious problem than today's interconnect lines. Further, as the interconnect lines are multi-layered, the shielding effects between neighboring lines makes the problem more complicated [47].

4 Crosstalk Noise Modeling of IC Interconnects

4.1 Introduction to the Crosstalk Noise

Today's high performance VLSI processes integrate more than several million transistors in an IC using deep submicron lithography. Increased chip size as large as 2 cm by 2 cm or more is being realized. These circuits are switching in less than a nano-second, which means both device and parasitics have several GHz bandwidth. Such technologies will keep improving in the future. Thus, the chip size and density keep increasing and the minimum feature size continues to decrease. In such high-speed and high-density VLSI circuits, crosstalk noise is one of the significant problems. This noise is due to electromagnetic coupling between interconnect lines[50][51]. In general, crosstalk noise has been accurately modeled by using the transmission

line theory [52]-[54] which can readily formulate the inductive, resistive, and conductive as well as capacitive coupling. While the model is very accurate, not only does it require an impractical amount of simulation time but also the direct mathematical analysis of coupled interconnect lines is exceedingly difficult. Therefore, most large scale IC CAD tools can not support such distributed circuit models. In order to improve the accuracy of CAD tools such as router or timing verification tools, a simple but accurate closed-form model for the crosstalk noise should be included within them since the analog simulation CAD tools are too slow for million transistor circuit analysis. Thus, lossless capacitive coupling is the standard as well as simple way of modeling multi-conductor interconnect coupling in the various CAD tools for the signal integrity verification.

4.2 Crosstalk Noise Model in Two RC-Coupled Lines

In the CMOS on-chip crosstalk noise analysis, dielectric loss (G) and inductance (L) can be neglected in the first order approximation although the inductance can not always be neglected [24]. The impedance conditions (i.e., source impedance R_s, the line resistance R_{line}, and load impedance, Z_L) have a substantial influence on the crosstalk noise. If $R_{line} > 2R_s$, the inductance effect can be neglected within 10% error. However, if $R_s >> R_{line}$ and $R_{line} >> Z_L$, the inductive coupling noise cannot be neglected and may become significant. However, this is not the case for the most of practical CMOS circuits because a driver resistance is not so large and a receiver is a capacitive load which has a large impedance. Thus the inductive coupling noise in CMOS circuits can be neglected as a first order approximation. Then, the circuit can be modeled as

$$\frac{\partial V^2(x,t)}{\partial x^2} = RC\frac{\partial V(x,t)}{\partial t}. \tag{71}$$

In the multi-conductor system, the signals and parameters can be presented in matrix form. In many papers [17][55]-[57], the equations were rigorously solved for two coupled lines. These solutions are very accurate because they are based on rigorous physical and mathematical analysis. However, for the techniques, time-consuming inverse Fourier transforms or convolution integrals are necessarily required to predict the time-domain responses since the solution equations are frequency-domain functions or time-domain convolution integrals. Moreover, they require many other computations and matrix manipulations. Hence, these approaches cannot simulate interconnect lines for the complicated circuit designs with thousands of transmission

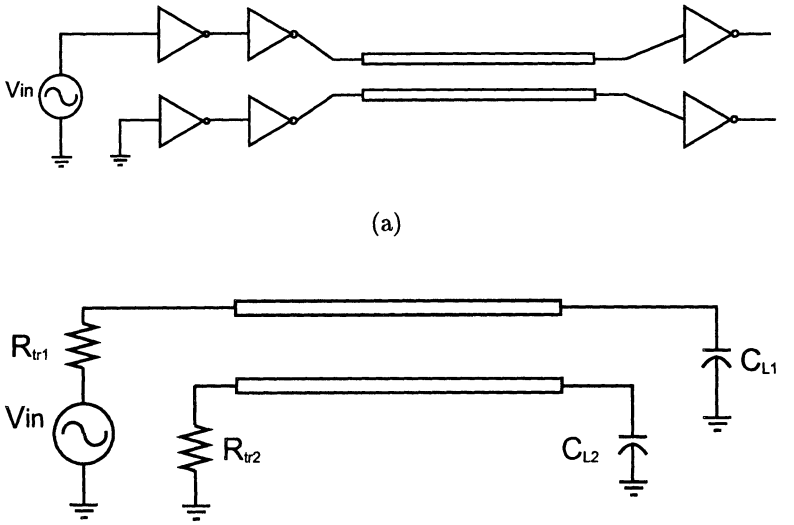


Figure 20. Schematic circuits composed of CMOS inverters and interconnects. The driving ports are modeled as resistance and the driven ports are modeled as capacitances, respectively.

lines and multiple interconnect layers. A simple but accurate model is required for such complicated VLSI circuit design.

The most basic building block in CMOS circuits is the inverter. A more complicated circuit analysis can be achieved by modifying the inverter circuits. Although the inverter circuits are a combination of non-linear devices, an inverter can be approximately modeled as a resistance in the driving stage of interconnect and a capacitance in its driven stage as shown in Figure. 20 [58]. The resistance for a moderate size inverter ($W/L >> 1$) ranges between 40Ω and 400Ω. Under such assumption, Sakurai derived a good interconnect crosstalk model [3]. He modeled the transmission line as an RC network and presented its step response as a power series [4]. Since the series is too complicated to be analytically solved, he used a first order approximation and then extended the simple expression to two coupled lines. That is, solving the wave equations which govern two capacitively coupled RC lines, Sakurai derived the crosstalk model in a closed form. The model is given by

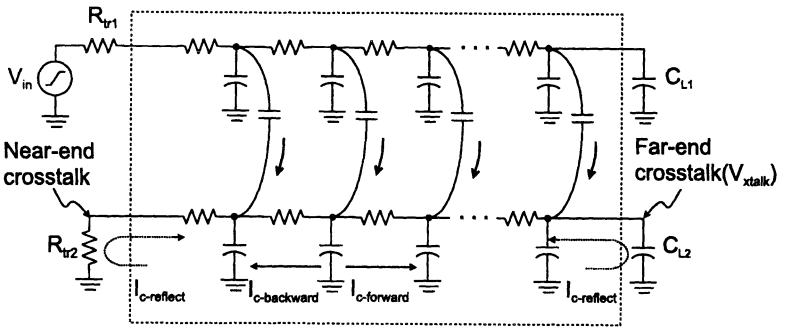


Figure 21. Physical coupling mechanism through the distributed RC network of coupled lines.

$$V_{xtalk}(t) = \frac{K_1}{2}\left[exp\left(-\sigma_1 \frac{t}{RC}\right) - exp\left(-\sigma_1 \frac{t}{RC + 2RC_c}\right)\right]. \qquad (72)$$

The K_1 and σ_1 are equal to (41) and (42) of the section 3. This model is good for its intended applications such as two lines and high input impedance gates. However, it overestimates or underestimates the amount of crosstalk signal for more general structures. Moreover, there is a need to extend the model to more than two coupled lines.

4.3 General Crosstalk Noise Model in RC-Coupled Lines

4.3.1 Effective Capacitance and Effective Resistance

The amount of crosstalk at the quiet line is strongly influenced by the termination conditions and transmission line parameters. A capacitive coupling current at the quiet line is divided into two parts (forward current wave and backward current wave) as shown in Figure. 21. Then these waves are reflected whenever the impedance is changed (i.e., at the driver and at the receiver) and their directions are reversed. When the noise pulse arrives at the far end, it approximately doubles because of the high impedance capacitance ($R_{line} << Z_L$). A large amount of the backward crosstalk (near end crosstalk) is reflected from the near end to the far end because of $R_s << R_{line}$ and adds up with the forward corsstalk. For a long line (in which a rise time(t_{rise}) is less than the round trip delay(t_{delay}) of the wave), the near end crosstalk voltage is independent of the line length but depends on the input driving voltage. In contrast, the far end crosstalk is proportional to the slope of driving signal and the length of the coupled

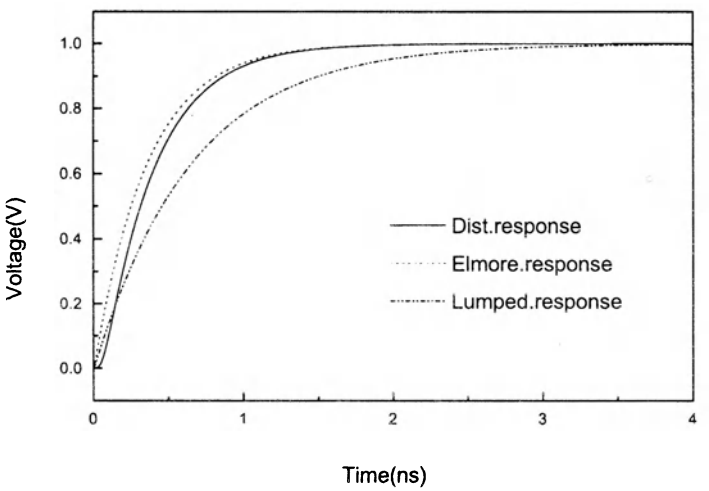


Figure 22. Step response of lumped and distributed interconnect models. The Elmore model shows good approximation of distribution model.

line. Thus, the longer the line length, the bigger the forward noise (far-end) buildup. Furthermore, far-end crosstalk has the wider noise signal than that of the near end while the near end crosstalk noise is a sharp spike. Thus, the worst-case crosstalk noise for practical CMOS circuit interconnects is considered to be generated at the far end. In addition, although signal coupling is due mainly to the coupling capacitance, other parameters such as the self-capacitance and self-resistance of the interconnect lines are strongly related to signal propagation speed, risetime, and signal coupling. Thus, the effective self-resistance and self-capacitance must be reconsidered. A simple lumped interconnect model does not match the transmission characteristics of long length of integrated circuit interconnects. In fact, the distributed model for signal propagation on single transmission line shows much a faster risetime than that of the lumped model. This can be shown in the SPICE simulation in Figure. 22. The distributed model shows a more rapid charging or discharging than the lumped model. The distributed model has a shorter time constant than that of the simple lumped model.

In the distributed circuits of the single line, the time delay is not a closed-form solution but it is bounded (even for the uniform RC interconnects). Nonetheless, in order to take the distributed transmission line effect of the interconnect line into account, Wyatt gives the approximated step response

as follows [5][49]

$$V_N(t) = V_{dd}(1 - exp(-t/T_{D-dist})) \tag{73}$$

where the T_{D-dist} is not a lumped-line time constant but a distributed-line time constant. The T_{D-dist}'s first order approximation is an Elmore time constant [4] which is a dominant pole approximation for an RC network. The Elmore time constant (T_{Elmore}) of an N-segment interconnect line is given by

$$T_{Elmore} \approx T_{D-dist} = \sum_{i=1}^{N} R_i \sum_{j=i}^{N} C_j = R_j C_j \frac{N(N+1)}{2} \tag{74}$$

where R_j and C_j are the line self-resistance and line self-capacitance of a segment, respectively. Thus, the distributed RC-interconnect time constant (T_{D-dist}) for a long line is totally different from the lumped RC-interconnect time constant ($T_{D-lump} = (R_j N)(C_j N) = RC$). In general, the distributed phenomena of the RC network can be well described by 10-segment model in both a phase and a magnitude (i.e., $N = 10$) [1]. Since the crosstalk voltage is a strong function of the self-capacitance and self-resistance, the lumped interconnect model parameters can be modified by using the Elmore time constant which can model fairly well the distributed phenomena of the RC line. That is, assuming the contribution to the system delay of the interconnect self-resistance and self-capacitance is identical and N is greater than 10, (74) can be represented by

$$T_{D-dist} \approx R_j C_j \frac{N(N+1)}{2} \approx \left(\frac{R_j N}{\sqrt{2}}\right)\left(\frac{C_j N}{\sqrt{2}}\right) \approx \left(\frac{R}{\sqrt{2}}\right)\left(\frac{C}{\sqrt{2}}\right). \tag{75}$$

Thus, the effective self-resistance and self-capacitance of the distributed transmission line are considered as

$$R_{eff} \equiv \frac{R}{\sqrt{2}} \quad \text{and} \quad C_{eff} \equiv \frac{C}{\sqrt{2}}. \tag{76}$$

That is, in (76), R_{eff} and C_{eff} can be regarded as the effective lumped values that model the distributed transmission line.

4.3.2 Generalized Crosstalk Model Using R_{eff} and C_{eff}

Including the previously derived effective interconnect resistance and capacitance, the lumped equivalent-network becomes Figure. 23. For the time

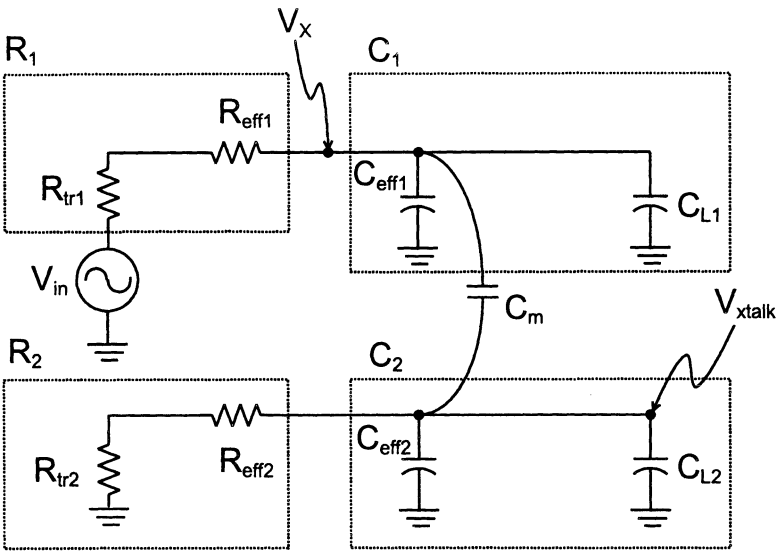


Figure 23. Coupled interconnect model wih an effective transmission line. The resistance and capacitance are grouped for the analysis with the effective transmission line parameter.

being, we assume the symmetrical structure with the identical input resistances of $R = R_1 = R_2$ and identical output loads of $C = C_1 = C_2$. The general model will be derived later. Then, solving the network equations under symmetrical conditions, the modeled crosstalk of the two coupled lines with effective transmission parameters will yield

$$V_{xtalk}(t) = \frac{1}{2}\left[exp\left(-\frac{t}{\left(R_{tr}+\frac{R}{\sqrt{2}}\right)\left(2C_m+\frac{C}{\sqrt{2}}+C_L\right)}\right)\right] - \left[exp\left(-\frac{t}{\left(R_{tr}+\frac{R}{\sqrt{2}}\right)\left(\frac{C}{\sqrt{2}}+C_L\right)}\right)\right] \quad (77)$$

where R_{tr}, R, C_m, C, and C_L are a driver resistance, an interconnect self-resistance, a coupling capacitance, an interconnect self-capacitance, and a load capacitance, respectively. If the signal path is very short, its crosstalk is not dominated by interconnect lines but by gate performances. For the short lines, the effective resistance and capacitance become meaningless and the transistor should be more accurately modeled considering its threshold voltage [49]. However, for such short lines, nobody is interested in crosstalk.

In contrast, for moderate length transmission lines or the long transmission lines such as more important critical path analysis, interconnect effects will dominate the total switching response.

The previous model can be readily extended to multiple line interconnect structures. In the multiple lines, the resistances of the interconnects are approximately equal to those of the single line if their cross sections are identical. However, the multiple line self-capacitances are not as simple as a single line. Although their cross sectional structures are exactly identical, the center line self-capacitance and outer line self-capacitance are not equal because of different electric field distributions. Moreover, the center line acts as shielding material for the outer line coupling. This new model is always efficient even if the capacitance is not symmetrical. Although the physical configurations and their terminations are different, the model can be modified for such structures. In order to derive a general model, we consider the i-th and j-th line of multiple coupled lines where the self-capacitances and load capacitances of i-th line and j-th line are not equal to each other, i.e.,

$$C_{ii} \neq C_{jj} \qquad \text{and} \qquad C_{Li} \neq C_{Lj}. \tag{78}$$

Letting the effective self-capacitances with load capacitance be

$$C_x = \frac{C_{ii}}{\sqrt{2}} + C_{Li}, \quad C_y = \frac{C_{jj}}{\sqrt{2}} + C_{Lj}, \text{ and } C_m = C_{ij} \tag{79}$$

and letting the effective self-resistance with transistor on-resistance be

$$R_x = \frac{R_{ii}}{\sqrt{2}} + R_{tr-i} \quad \text{and} \quad R_y = \frac{R_{jj}}{\sqrt{2}} + R_{tr-j}. \tag{80}$$

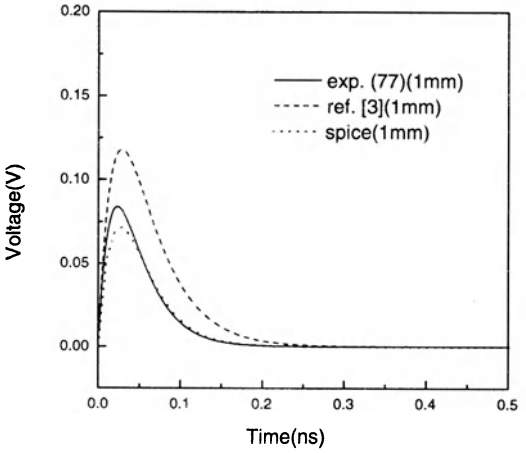
The crosstalk voltage $V_{xtalk}(t)$ for the i-th and j-th line can be derived as follows

$$V_{xtalk}(t) = \frac{C_m}{C_{eq}} \left[exp(-\beta t) - exp(-\alpha t)\right] \tag{81}$$

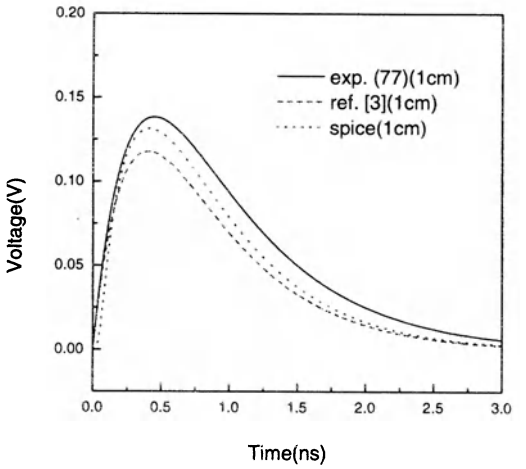
where the parameters are

$$C_{eq} = \sqrt{\left(\frac{R_x}{R_y}\right)^2 (C_x + C_m)^2 + (C_y + C_m)^2 + 2\left(\frac{R_x}{R_y}\right)\left((C_m)^2 - C_x C_y - C_x C_m - C_y C_m\right)} \tag{82}$$

$$\alpha = \frac{b + \sqrt{b^2 - 4a}}{2a} \quad \text{and} \quad \beta = \frac{b - \sqrt{b^2 - 4a}}{2a} \tag{83}$$



(a) 1 mm long line



(b) 1 cm long line

Figure 24. Two coupled line crosstalk simulation with HSPICE and interconnect model [3]. ($C_c = 0.4pF/cm, C_{self} = 0.87pF/cm, R_S = 50m\Omega/\square$).

$$a = R_x R_y \left(C_x C_y + C_x C_m + C_y C_m\right) \tag{84}$$

and

$$b = R_x \left(C_x + C_m\right) + R_y \left(C_y + C_m\right). \tag{85}$$

However, if the driver resistances of the i-th and j-th line are equal to each other (i.e., $R = R_x = R_y$) and $C_x = C_y$ (i.e., symmetrical structure), the expression reduces to the simpler expression of (77). In general, for the multiple lines, the superposition principle can be applied to the multiple sources. Thus, if the i-th line has signal source and j-th line is victim line, all the parameters of i-th and j-th line are presented with superscripts as $A_j \equiv C_m^{ij}/C_{eq}^{j}$. Then

$$V_{xtalk}^{j}(t) = A_j \left[exp\left(-\beta_j t\right) - exp\left(-\alpha_j t\right)\right]. \tag{86}$$

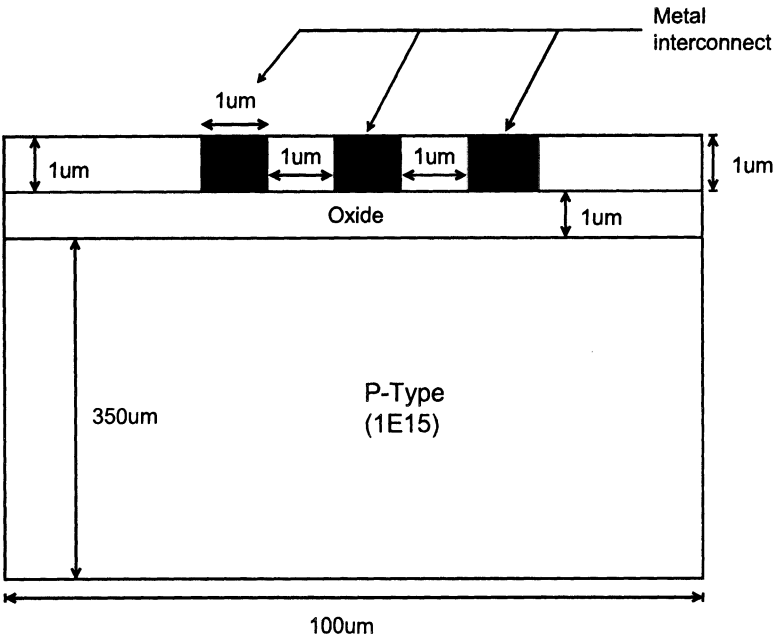
Hence, the total crosstalk voltage with k-independent signal sources for n-multiple line systems becomes [78],

$$V_{xtalk}^{total}(t) = \sum_{j=1}^{k} A_j \left[exp\left(-\beta_j t\right) - exp\left(-\alpha_j t\right)\right]. \tag{87}$$

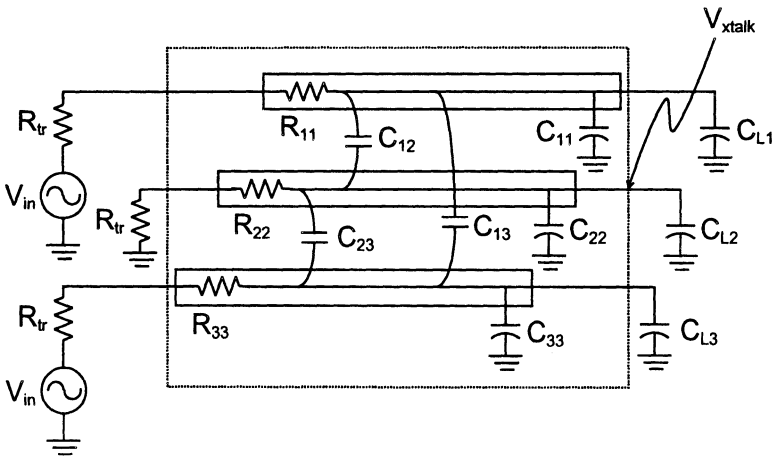
The above (87) can be applied to general multiple lines and multiple source interconnect systems.

4.3.3 Simulation Examples with the Model

For a transmission line modeled as an RC network, a 10 segment RC-ladder network can be accurate in both magnitude and phase [1]. Thus, the previous model has been compared with a 10 segment RC-ladder SPICE model. Note, the analytic model has been derived under the assumption that input excitation is the unit step function. Intuitively, the sharper risetime induces more signal coupling and crosstalk. For the simulation and verification of the model, i.e., expression (87), triple transmission lines are defined as a test circuit. Triple interconnect lines are the most widely used test circuits to predict circuit failures due to crossstalk. Although the model (87) can be applied to heterogeneous structures, we assume that all the line structures have identical conductors in order to compare with other models. The triple lines with the same layout width and length are shown in Figure. 25. The capacitance parameters are calculated by using MEDICI[81] for lines that are 1cm long. Transistor resistances are assumed as 82.7Ω and load



(a) Cross section of the triple coupled lines



(b) Circuit model

Figure 25. Equivalent circuit model of triple-coupled lines for model verification. The far end of the center line is the test point for the crosstalk.

Items / Interconnect structures	SPICE		Model		%Error	
	1cm	5mm	1cm	5mm	1cm	5mm
Normal triple	0.131	0.118	0.138	0.129	+5.3	+9.3
Thick triple	0.339	0.311	0.359	0.341	+5.9	+9.6
Narrow triple	0.401	0.365	0.416	0.395	+3.7	+8.2
5 line as 3rd line victime	0.276	0.247	0.290	0.269	+5.1	+8.9
5 line as 2nd line victime	0.254	0.230	0.270	0.252	+6.3	+9.6

Table 1. % error comparison of model with SPICE simulation.

capacitances are simply modeled as the gate capacitance of 76 fF. For the structure, the sheet resistance of the metal is assumed as $R_S = 50m\Omega/\square$ and triple line capacitances$[pF/cm]$ are as follows

$$[\mathrm{C}] = \begin{pmatrix} 0.850 & 0.400 & 0.453E-3 \\ 0.400 & 0.623 & 0.400 \\ 0.453E-3 & 0.400 & 0.850 \end{pmatrix}$$

The on-diagonal elements represent the self-capacitance of each line, C_{kk}. The off-diagonal elements show the coupling capacitance between each lines, C_{jk}. Using these example values, the model(87) and SPICE show an excellent agreement as shown in Figure. 26. That is, the model is within 10% error of the SPICE simulation in the worst case. Moreover, unlike other models, the model never underestimates the response value. Therefore, there are no catastrophic failures due to the underestimation. The metal aspect ratio controls the interconnect resistance. Currently, the aspect ratio of the advance process technologies is larger than 1.22 [59]. For the case of the aspect ratio=2, SPICE simulation and the model shows very good agreement as summarized in Table 1 (thick triple). Furthermore, if the line spacing is a half of Figure. 25 and the line thickness is the same, the coupling capacitance becomes much larger than the self-capacitance. Even for this case, the model and SPICE simulation show good agreements as shown in Table 1 (narrow triple). In another example of the more general 5 line structures, the physical line dimensions and spaces are identical to the triple line structure of Figure. 25 However, the electrical field distribution around the interconnects are different from the triple line structure and the capacitance matrix for the 5-line structure$[pF/cm]$ is

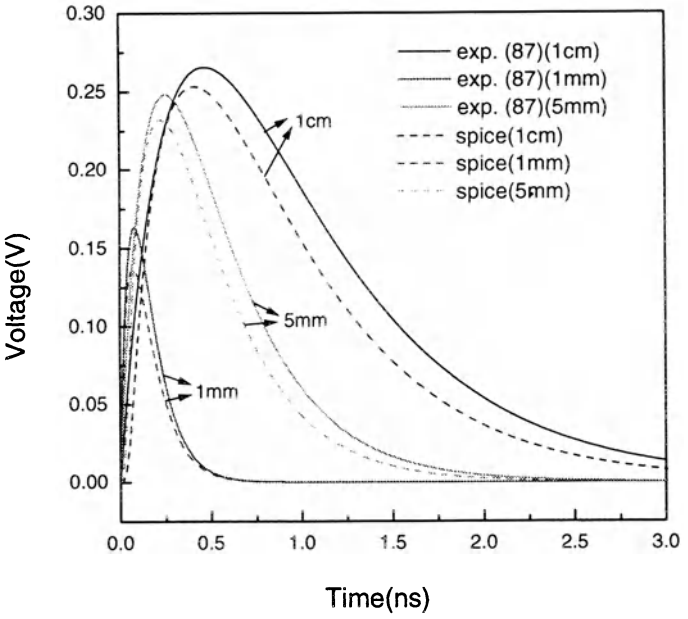


Figure 26. Crosstalk simulation for triple lines with two signal sources. The new model for triple interconnect lines show good agreements with HSPICE.

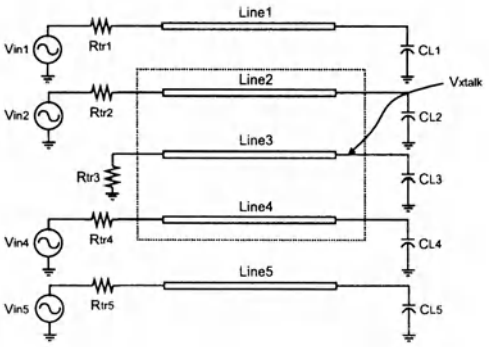
$$[C] = \begin{pmatrix} 1.100 & 0.410 & 0.416E-3 & 0.166E-5 & 0.654E-8 \\ 0.410 & 0.631 & 0.400 & 0.420E-3 & 0.166E-5 \\ 0.416E-3 & 0.400 & 0.630 & 0.400 & 0.416E-3 \\ 0.166E-5 & 0.420E-3 & 0.400 & 0.631 & 0.410 \\ 0.654E-8 & 0.166E-5 & 0.416E-3 & 0.410 & 1.100 \end{pmatrix}.$$

As we can see in the above capacitance matrix, the outer line self-capacitance and coupling capacitance are different from those of the inner lines. Thus, their crosstalk noise is clearly different from the triple lines. Moreover, since the different switching scenarios of input line sources may cause different crosstalk phenomena, each switching scenario must be investigated. The first case, the 3^{rd} line (center line) is the victim line and all other lines are switching. The equivalent circuits are shown in Figure. 27(a) and the signal transition of input sources is shown in Figure. 27(b). Note, because of the shielding effects the lines within the dotted box in Figure. 27(a) have a dominant effect on the crosstalk. The model and SPICE simulation for different lengths are compared with undergoing several switching scenarios. SPICE simulation shows the worst case crosstalk is the case_A_1. As a second example for the same structures, the 2^{nd} line is victim line and all other lines are switching as culprit lines shown in Figure. 28. The worst case crosstalk is the case_B_1 where all the lines except line 2 go from logic 0 to logic 1. For the simulation of different switching scenarios, SPICE simulation and crosstalk model are shown in Table 1. In summary, the other combinations of the switching scenarios have agreement that is identical to these two special cases. Thus the model can accurately estimate the crosstalk for general multi-line systems.

5 Simultaneous Switching Noise Modeling due to IC Packaging

5.1 Introduction to the IC Package Noise

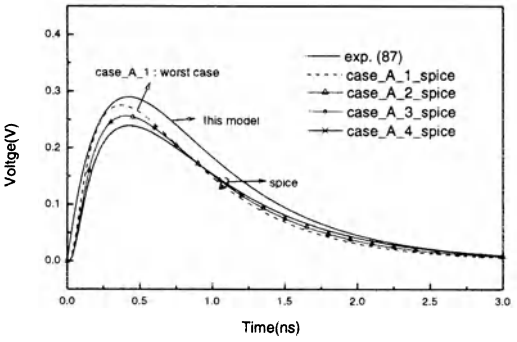
With the rapid improvement of semiconductor process technologies and circuit performances, today's VLSI system of several GHz clock bandwidth needs improved package design methodologies [50][51][61]. In such high-performance VLSI circuits, the most critical bottleneck due to the IC package is the reference potential fluctuation during circuit switching, i.e., simultaneous switching noise (SSN) [62]-[68]. Particularly, the SSN results in serious performance degradation and system failures due to the reduction of noise margin, the increase in the effective signal delay, glitches, and signal distortion. Since the noise is proportional to the number of switching gates



(a) Equivalent circuits for 5-line structure with the 3^{rd} line victim

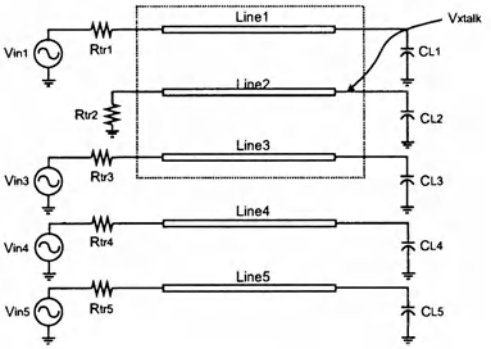
Case \ Input	Vin1	Vin2	Vin3	Vin4	Vin5
Case_A_1	0→1	0→1	0	0→1	0→1
Case_A_2	0	0→1	0	0→1	0→1
Case_A_3	0→1	0→1	0	0→1	0
Case_A_4	0	0→1	0	0→1	0

(b) Table for different swiching scenarios



(c) Model and SPICE for 1 cm long lines

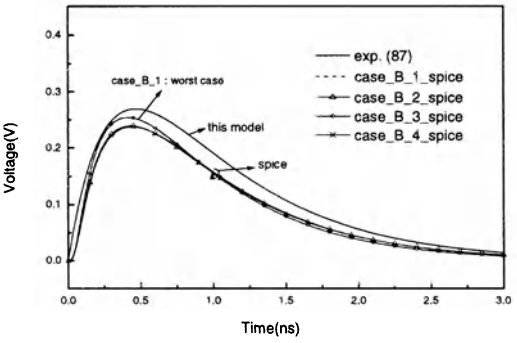
Figure 27. Model and SPICE simulation for 5 line structures; The 3^{rd} line victim with different switching scenarios.



(a) Equivalent circuits for 5-line structure with the 2^{rd} line victim

Case \ Input	Vin1	Vin2	Vin3	Vin4	Vin5
Case_B_1	0→1	0	0→1	0→1	0→1
Case_B_2	0→1	0	0→1	0	0→1
Case_B_3	0→1	0	0→1	0→1	0
Case_B_4	0→1	0	0→1	0	0

(b) Table for different swiching scenarios



(c) Model and SPICE for 1 cm long lines

Figure 28. Model and SPICE simulation for 5 line structures; The 2^{rd} line victim with different switching scenarios.

and the many package design parameters, circuit designers must take these packaging effects into account at the early phase of circuit design. Although there are many other problems concerned with IC packages, the SSN noise is here discussed since it is the most significant problem.

Many package design methodologies concerned with simultaneous switching noise (SSN) models have been developed [69]-[74]. Vaydianath et al. derived a good SSN model based on the long-channel MOS-transistor approximation that considered the negative feedback due to the voltage across the inductor [62]. Since the model was based on the long-channel approximation, their model could not fully predict the SSN for sub-micron-based circuits. The simple long-channel approximation clearly overestimates the SSN noise because the model predicts much more current variation than is presented in short channel devices. Unlike the long-channel model of [62], Vemuru [63] modeled the SSN based on Sakurai's alpha-power model (i.e., linear power law) [75] and similarly, Yang et al. also modeled the SSN by employing the similar linear power law of sub-micron-device drain current [72]. However, although they take the velocity saturation effects of sub-micron devices into account, the model did not consider the physical behaviors of transistor, inductance, and capacitance as a system. Particularly, the output load capacitor strongly affects the noise oscillation frequency, which is also a function of the current slew rate. Therefore a more rigorous physical interpretation of the transistor operation regions is necessary to estimate the accurate SSN and to design the advanced packaging. In this section, fundamental theory concerned with package noise is presented. Then a simple but accurate SSN model is introduced, followed by the package design meghodologies.

5.2 Circuit Component Effect Concerned with SSN

NMOS transistors are concerned with the discharging path of CMOS gates. Thus, the SSN in the ground path of CMOS circuits is closely related to the NMOS transistor operation mode. A MOS device experiences 3 operation regions during the switching: i.e., cut-off, saturation, and linear region (non-saturation region). Thus, the transistor output current is not constant during the whole transition of input gate signal. In the CMOS inverter circuit as shown in Figure. 29, the saturation time of the transistor is a function of device parameters and output load capacitance. Thus, without considering both the package inductance and capacitance, the overly simple saturation current modeling may result in an erroneous estimate of SSN.

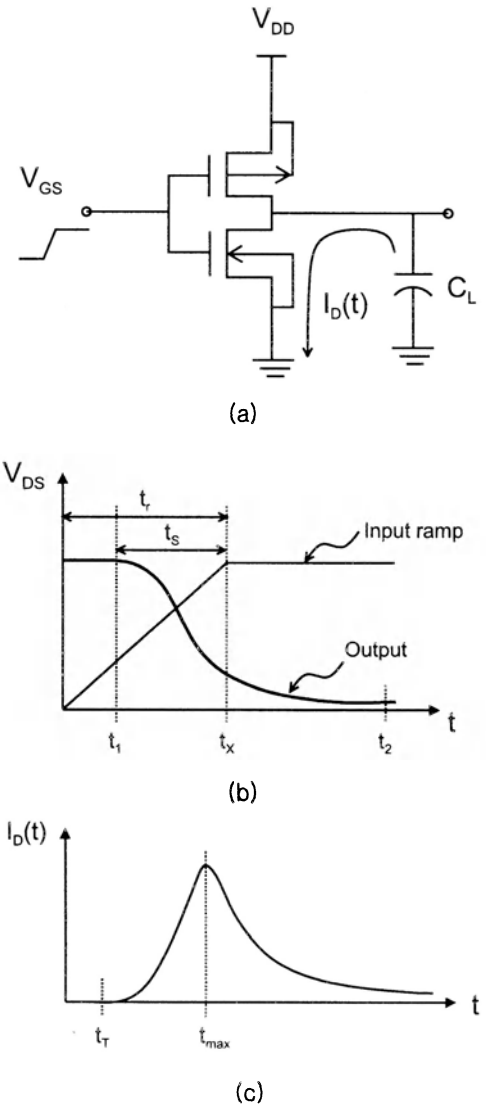


Figure 29. Signal transients in the input and output of a CMOS inverter (a)circuit diagram (b)signal transients of input ramp and output response (c)current flow through the NMOS device.

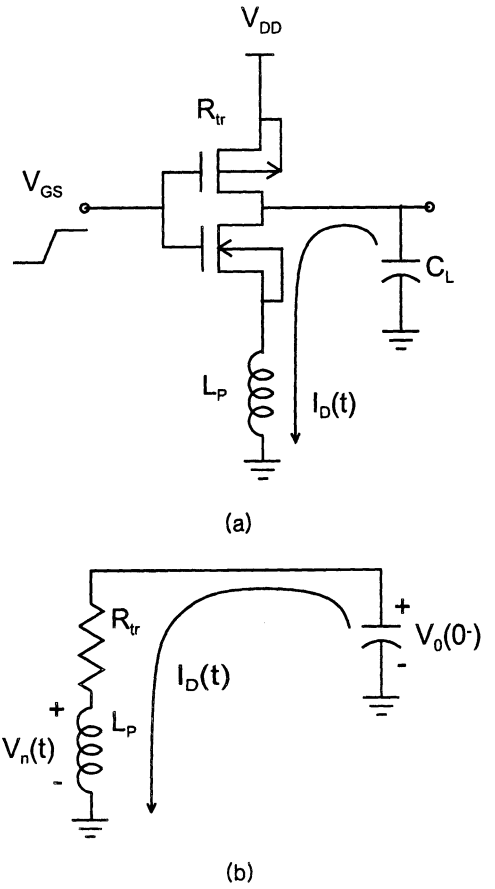


Figure 30. Equivalent swiching circuit model for the discharging path with a package(ground)inductance.

The current flow mechanism, including both inductance and transistor, can be approximately quantitatively described by modeling the transistor as a resistance. The CMOS inverter circuit is shown in Figure. 30(a). Modeling the NMOS device as a simple resistance during the transition, an equivalent circuit for a discharging path becomes an RLC circuit as shown in Figure. 30(b). In order to determine the current slew rate, the current must be found as a function of time. Then the time, t_{max}that the maximum current flows, can be evaluated. The current in such RLC circuits can be readily determined as follows [76][77]

$$I_D(t) = \frac{V_{DD}}{L_p} \frac{1}{\sqrt{\omega^2 - \alpha^2}} e^{-\alpha t} sin\left(\sqrt{\omega^2 - \alpha^2} t\right) \tag{88}$$

where

$$\omega \equiv \frac{1}{\sqrt{L_p C_L}} \quad \text{and} \quad \alpha \equiv \frac{R_{tr}}{2L_p}$$

where R_{tr} is a linearized transistor resistance model. However, in more detail, the solution of the current $I_D(t)$ is divided into three special cases depending upon transistor resistance, package inductance, and load capacitance. Now defining a critical resistance as

$$R_{cr} \equiv 2\sqrt{\frac{L_p}{C_L}}, \tag{89}$$

the solution may result in one of three cases, the over-damped case, the critically-damped one, or under-damped one. Since the $I_D(t)$ may have different forms for these three cases, the t_{max} must be determined separately for each case. That is, the time that makes the time derivative of each current equation be zero must be calculated.

Over-Damped Case ($R_{tr} > R_{cr}$) :

$$I_a(t) = \frac{V_{DD}}{L_p\sqrt{\alpha^2 - \omega^2}} e^{-\alpha t} sinh\left(\sqrt{\alpha^2 - \omega^2}t\right) \tag{90}$$

Then the time that maximum current flows can be determined by the first derivative of (90) as follows

$$t_{max-a} = \frac{1}{\sqrt{\alpha^2 - \omega^2}} ln\left(\frac{\alpha + \sqrt{\alpha^2 - \omega^2}}{\alpha - \sqrt{\alpha^2 - \omega^2}}\right). \tag{91}$$

Critical-Damped Case ($R_{tr} = R_{cr}$) :

$$I_b(t) = \frac{V_{DD}}{L_p} te^{-\omega t} \quad . \tag{92}$$

Thus the maximum current flows at the time of

$$t_{max-b} = \frac{1}{\omega} = \sqrt{L_p C_L}. \tag{93}$$

Under-Damped Case ($R_{tr} < R_{cr}$) :

$$I_c(t) = \frac{V_{DD}}{L_p\sqrt{\omega^2 - \alpha^2}} e^{-\alpha t} sin\left(\sqrt{\omega^2 - \alpha^2}t\right) \quad . \tag{94}$$

Similarly the maximum current flows at the time of

$$t_{max-c} = \frac{\pi}{2} \frac{1}{\sqrt{\omega^2 - \alpha^2}}. \tag{95}$$

Therefore, the time that the maximum current flows is a complicated function of load capacitance, ground line package inductance, and device parameters. It is noteworthy that many of submicron-technology-based I/O drivers may have smaller resistance than R_{cr}. Thus, in many cases, the ground path current is dominated by the under-damped case which is oscillating [77]. The time that sinusoidal current reaches the first instantaneous peak value with reasonable parameters is usually very short in all three cases, but it is a strong function of the load capacitance. If the under-damped ground path current has more than one sinusoidal peak value during the input transition time (i.e., input rise time), during the oscillation there may be a much larger current slew rate between the positive peak and the negative peak current. This large slew rate between the peak-to-peak currents results in a significantly larger SSN. This is a very important fact because the SSN due to the large peak-to-peak current slew rate may be more than twice those of [63][72] which simply assume that the ground current always has only one sinusoidal peak during the input transition time.

5.3 SSN Model due to the Package

The discharging current of the transistor is dependent on its operation mode, package inductance, and the load capacitance. The transistor current flow due to the transients of the input and output signal is schematically shown in Figure. 31. The t_r is defined as a time that an input ramp transits from 0 to V_{DD} and the t_T is defined as a time that the input ramp transits from 0 to NMOS transistor threshold voltage V_{tn}. The current flow mechanism of NMOS transistor can be qualitatively explained as follows. At the region 1 to 2, the current does not flow because the NMOS transistor does not operates until $V_{GS}(t) > V_{tn}$. Therefore, $V_{out}(t)$ will stay nearly at a constant value rather than it changing. In contrast, $V_{GS}(t)$ still goes up toward V_{DD}. Next, 2 to 3 transition is dominated by both transistor current and inductor voltage. Thus, there are many possible current paths as shown in Figure. 31. As mentioned, the current may not be uniformly changing during the transit time in this region. That is, the current slew rate near region 3 is larger than that near the region 2 because the amplifier (i.e., the inverter) has the largest voltage gain near the switching threshold voltage. Because of this fact, it cannot be simply assumed that the maximum current slew rate is equal to an average slew rate between t_T and t_r. In the next region, that is, the 3 to 4 transition, the linear region where the transit time is not as fast as the saturation region. Note, however, because of the package inductance and load capacitance, the ground path current may oscillate.

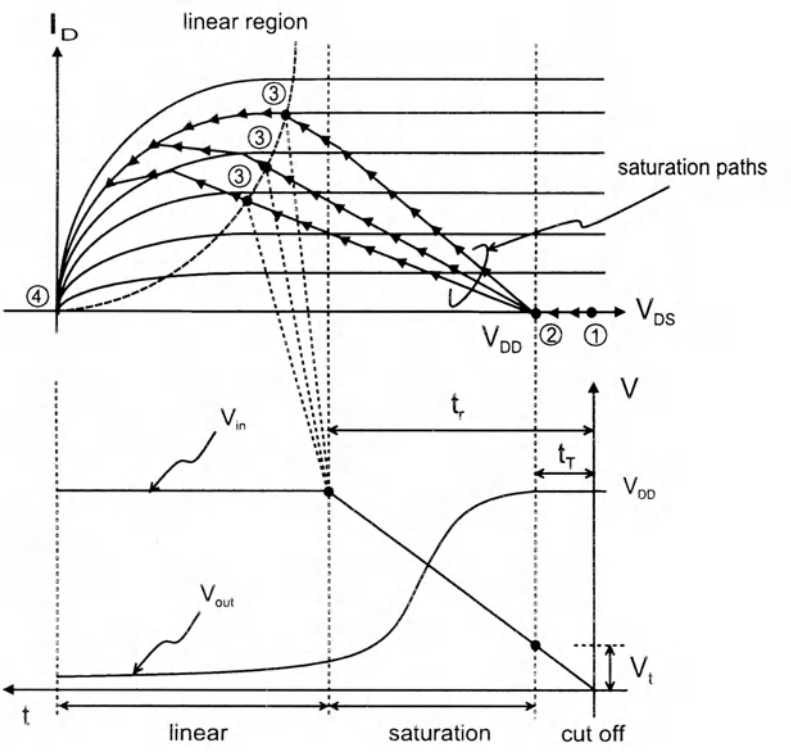


Figure 31. Schematic output current flow during the input and output voltage transients.

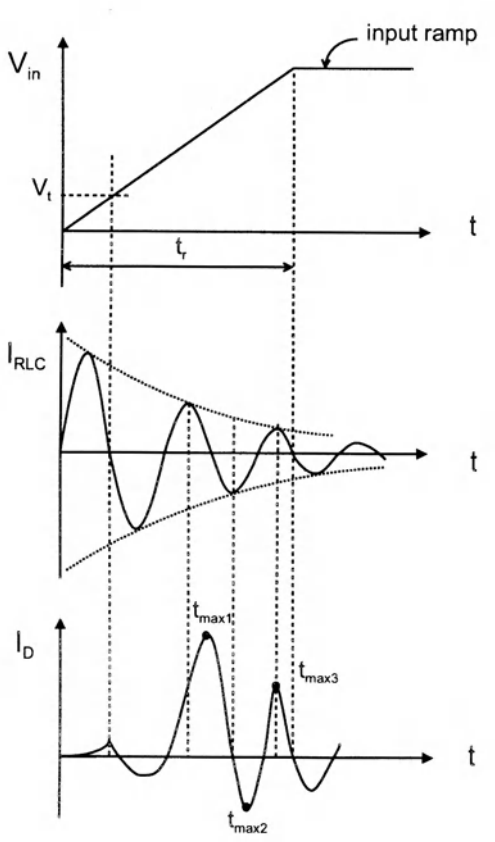


Figure 32. Schematic diagram of input signal transient and the oscillating currents of RLC circuit(I_{RLC}) and NMOS device switching current(I_d).

If the ground path current oscillates, there may be many occasions where the current varies rapidly within a very short time as shown in Figure. 32. This results in a large current slew rate, even if the transistor is in the linear operation region. For the SSN analysis, all these effects are taken into account. Further, today's sub-micron device drain current is not dominated by the conventional long-channel-based square power but by the alpha-power due to the velocity saturation effect [75]. The alpha of the today's deep sub-micron device current is nearly one [63]. Considering all these effects, the simultaneous switching noise was developed by Eo et al.[79],

$$V_{nmax} \approx \frac{nkL_p\beta_n}{2kt_r + 2n\,(k-1)\,L_p\beta_n} V_{DD}. \tag{96}$$

Herein, β_n is the device transconductance value near the switching threshold voltage (V_{switch}) of the gate. Note that the dimension of β_n must not be considered as $[A/V^2]$ but $\left[A/V^{alpha}\right]$. The k is a constant factor to adjust the maximum current slew rate. If the k is simply assumed as 2, it is similar to the conventional model. However, the intersection of the time axis of the current slope between t_{switch}(the time corresponding to V_{switch}) and t_r is not at $t = 0$ but at some point greater than t_T. A reasonable k value for most circuit operations is about 3 to 4. In reality, it can be readily extracted for a particular technology once a simulation is performed. If the k is assumed as 3 in order to maintain the model in a simple analytic form

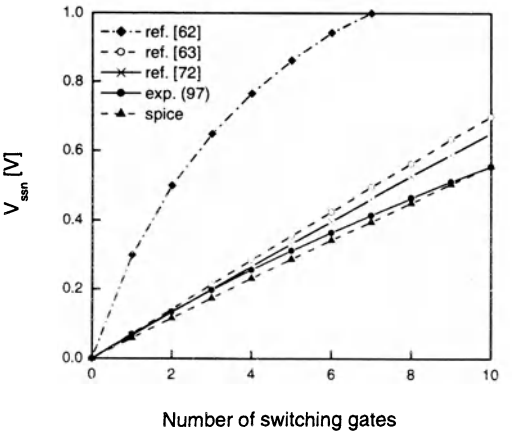
$$V_{nmax} \approx \frac{nL_p\beta_n}{2t_r + 1.3nL_p\beta_n}V_{DD}. \tag{97}$$

In order to show the accuracy of model (97), the simulation was performed with 0.35um CMOS process-based circuits. The simulation employed the level 49 MOS model [BSIM3] of HSPICE. The NMOS device transconductance for the simulation is $\beta_n \approx 43mA/V^{alpha}$. As shown in Figure. 33 and Figure. 34, the model (97) has excellent agreements for various package design parameters with HSPICE simulation results within about 5% error.

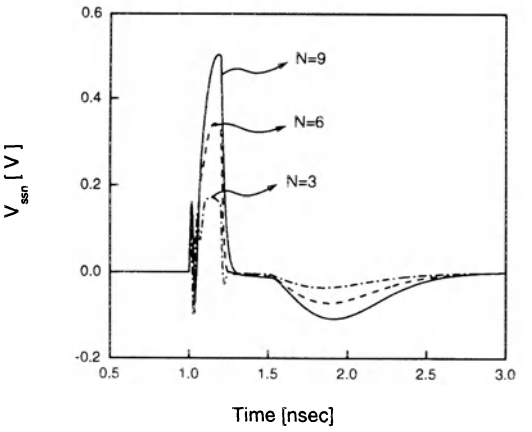
5.4 Design Consideration

The SSN variation is a strong function of inductor, load capacitor, and transistor size. Since the current slew rate maximum happens near t_{switch} to t_r for practical circuits, the slew rate and transistor β_n must be determined near the t_{switch} to t_r. In addition, the capacitance size is quite important because it is a selectable design parameter unlike other fixed parameters, such as the inductor, which is dependent on the package type and ground placement within a chip. Particularly, a small load capacitor may cause the under-damped case (damped-oscillating case) by making R_{cr} too large. The best design guide lines are as follows. Since the inductance is strongly related with physical structures, such as package type or ground configuration, the package type with the minimum inductance should be selected. In this case, the driver size should be large because the resistance is inversely proportional to the driver size. The transistor resistance must be calculated by using the alpha-power law as follows:

$$R_{tr} \approx \frac{V_{DD}}{\frac{\beta_n}{2}(V_{GS} - V_{tn})^{alpha}}. \tag{98}$$

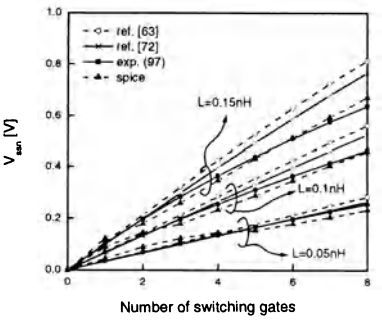


(a) SSN noise viariations for the rise time=0.2nsec and L_p=0.2nH

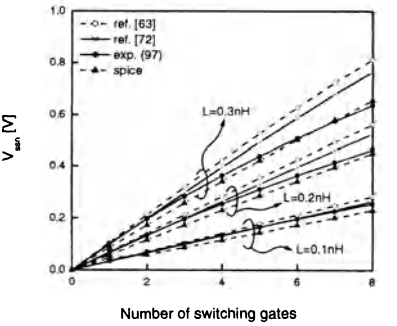


(b) SPICE-simulation-based time domain waveforms of the SSN

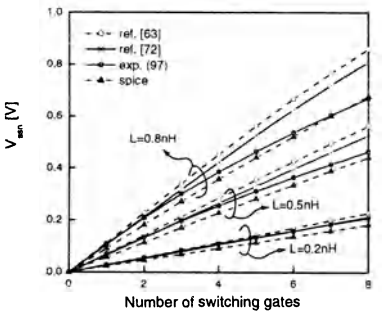
Figure 33. Verification of the SSN models with SPICE simulation results for the driver size of $\beta_n = 43mA/V^{alpha}$(i.e., $(W/L)_{nmos}$=(80/0.35)).



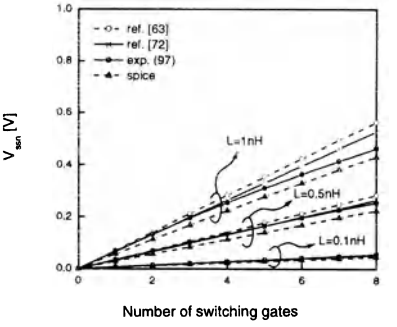
(a) SSN variations for L_p=0.05nH, 0.1nH and 0.15nH when tr=0.1nsec



(b) SSN variations for L_p=0.1nH, 0.2nH and 0.3nH when tr=0.2nsec



(c) SSN variations for L_p=0.2nH, 0.5nH and 0.8nH when tr=0.5nsec



(d) SSN variations for L_p=0.1nH, 0.5nH and 1nH when tr=1nsec

Figure 34. SSN in terms of various package design parameters(i.e. the rise time, the package inductance, and the number of the switching gates). Note that the driver size is $(W/L)_{pmos}$=(150/0.35) and $(W/L)_{nmos}$=(80/0.35).

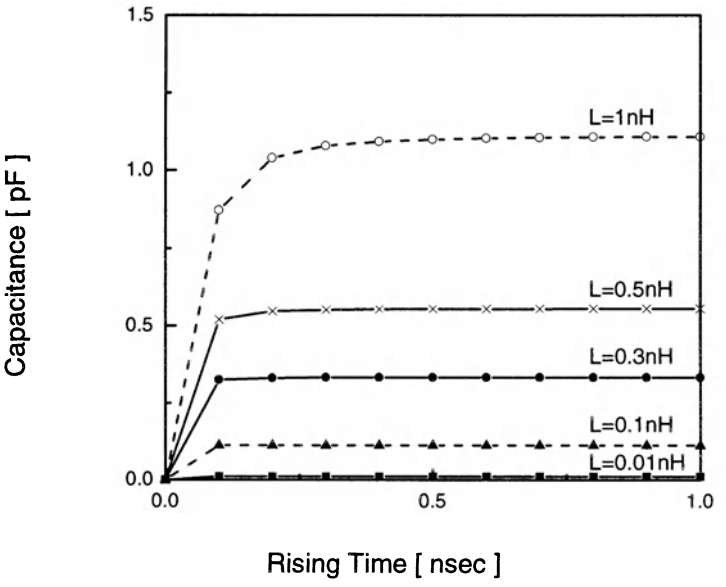


Figure 35. Minimum load capacitance(C_L) requirements for different inductances(L_P) and rise times to meet the minimum SSN.

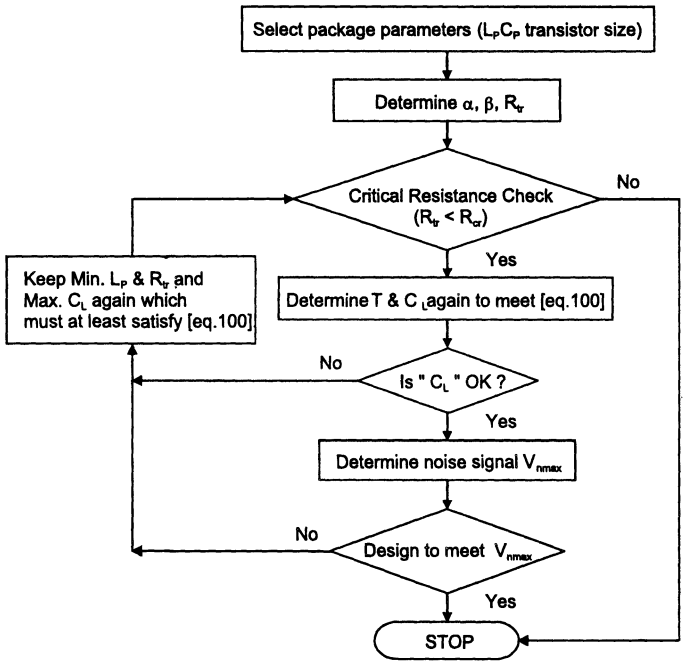


Figure 36. Design procedures for optimal IC package design.

If the noise is still oscillating, the load capacitance must be increased. Once the resistance is determined, a quarter of the current oscillation period must be greater than t_r. Thus the capacitance must satisfy

$$\frac{T}{4} = \frac{\pi}{\sqrt{(L_p C_L)^{-1} - \alpha^2}} \geq t_r. \tag{99}$$

Rearranging (99), the load capacitance should meet

$$C_L \geq \left[L_p \left(\left(\frac{\pi}{2t_r} \right)^2 + \left(\frac{R_{tr}}{2L_p} \right)^2 \right) \right]^{-1} \tag{100}$$

Thus (100) gives the minimum load capacitance for non-oscillating within an input rise time for a given R_{tr} and a given L_p. Alternatively, if the load capacitance C_L is given, the inductance L_p must meet (100). Given the device and inductance, the load capacitance requirements with the rise time variations are shown in Figure. 35. Clearly, for a large inductance, the capacitance size must be increased. The design procedures are summarized in Figure. 36.

Note that expression (97) and expression (100) are extremely valuable design equations because they can provide the circuit designers with an accurate design methodology to meet their design goal as well as insight into SSN and I/O device size. That is, circuit designers can accurately predict the SSN noise of their circuits by using (97) and therefore improve upon their design by using both (97) and (100).

Acknowledgments
The author would like to thank his student, Mr. Woojin Jin who proofread the manuscript as well as prepared many useful figures.

References

[1] R. J. Antinone and Gerald W. Brown, *"The modeling of Resistive Interconnects for Intergrated Circuits,"* IEEE Journal of Solid-State Circuits, vol. SC-18, pp. 200-2-3, Apr. 1983.

[2] T. Sakurai and K. Tamaru, *"Simlpe Formula for Two-and Three-Dimensional Capacitances,"* IEEE Trans. Electron Device, vol. ED-30, No.2, pp. 183-185, Feb. 1983.

[3] T. Sakurai, *"Closed-Form Expressions for Interconnection Delay, Coupling, and Crosstalk in VLSI's,"* IEEE Trans. ED., vol. 40, No. 1, pp. 118-124, Jan. 1993.

[4] W. C. Elmore, *"The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers,"* Journal of Applied Physics, vol. 19, pp. 55-63, Jan. 1948.

[5] J. Rubinstein, P. Penfield, Jr., and M. Horowitz, *"Signal Delay in RC Tree Networks,"* IEEE Trans. CAD., vol. CAD-2, No. 3, pp. 202-211, July 1983.

[6] H. R. Kaupp, *"Waveform Degradation in VLSI Interconnections,"* IEEE JSSC, vol. 24, pp. 1150-1153. Aug. 1989.

[7] A. Deng and Y. Shiau, *"Generic Linear RC Delay Modeling for Digital CMOS Circuits,"* IEEE Trans. CAD., vol. 9, No. 4, pp. 367-376. Apr. 1990.

[8] T. Sakurai, *"Approximation of Wiring Delay in MOSFET LSI,"* IEEE JSSC., vol. SC-18, No. 4, pp. 418-426, Aug. 1983.

[9] A. B. Kahng and S. Muddu, *"An Analytical Model for RLC Interconnects,"* IEEE Trans. CAD., Vol. 16, No. 12, pp. 1507-1514, Dec. 1997.

[10] Y. Massoud et al., *"Layout Techniques for Minimizing On-Chip Interconnect Self Inductance,"* in Proc., ACM/IEEE Design Automation Conf., 1998, pp. 566-571.

[11] B. Krauter and S. Mehrotra, *"Layout Based Frequency Dependent Inductance and Resistance Extraction for On-Chip Interconnect Timing Analysis,"* in Proc., ACM/IEEE Design Automation Conf., 1998, pp. 303-308.

[12] Y. I. Ismail, E. G. Friedman, and J. L. Neves, *"Figure of Merit to Characterize the Importance of On-Chip Inductance,"* in Proc., ACM/IEEE Design Automation Conf., 1998, pp. 560-565.

[13] S. Lin and E. Kuh, *"Transient Simulation of Lossy Interconnects Based on the Recursive Convolution Formulation,"* IEEE Trans. CAS, vol. 39, pp. 879-892, Nov. 1992.

[14] L. T. Pillage and R. A. Roher, *"Asymptotic Waveform Evaluation for Timing Analysis,"* IEEE Trans. CAD, vol. 9, No. 4, pp. 352-368, Apr. 1990.

[15] T. S. Blazeck and R. Mittra, *"Transient Analysis of Lossy Multiconductor Transmission Lines in Nonlinear Circuits,"* IEEE Trans. CPMT., vol. 14, no. 3, pp. 618-627, Sep. 1991.

[16] Y. Eo and W. R. Eisenstadt, *"High-Speed VLSI Interconnect Modeling Based on S-Parameter Measurements,"* IEEE Trans. CPMT., vol. 16, No. 5, pp. 555-562, Aug. 1993.

[17] Y. Eo and W. R. Eisenstadt, *"Simulation and Measurements of Picosecond Signal Transients, Propagation, and Crosstalk on Lossy VLSI Interconnect,"* IEEE CPMT. Part A, vol. 18, No. 1, pp. 215-225, Mar. 1995.

[18] H. Guckel et al., *"A parallel-Plate Waveguide Approach to Microminiaturized, Planar Transmission Lines for Integrated Circuits,"* IEEE Trans. MTT., vol. 15, pp. 468-476, Aug. 1967.

[19] H. Hasegawa, M. Furukawa, and H. Yanai, *"Properities of Microstrip Line on Si-SiO_2 System,"* IEEE Trans. MTT, vol. MTT-19, pp. 869-881, Nov. 1971.

[20] R. Sorrentino, G. Leuzzi, and A. Slbermann, *"Characteristics of Metal-Insulator-Semiconductor Coplanar Waveguides for Monolithic Microwave Circuits,"* IEEE Trans. MTT, vol. 32, no. 4, pp. 410-416, Apr. 1984.

[21] Y. Fukuoka. Y. Shih, and T. Itoh, *"Analysis of Slow-Wave Coplanar Waveguide for Monolithic Integrated Circuits,"* IEEE Trans. MTT., vol. 31, No. 7, pp. 567-573, Jul. 1983.

[22] T. Shibata and E. Sano, *"Characterization of MIS Structure Coplanar Transmission Lines for Investigation of Signal Propagation in Integrated Circuits,"* IEEE Trans. MTT., vol. 38, No. 7, pp. 881-890, Jul. 1990.

[23] E. Groteluschen, L. S. Dutta, and S. Zaage, *"Quasi-Analytical Analysis of the Broadband Properties of Multiconductor Transmission Lines on Semiconducting Substrates,"* IEEE Trans. CPMT-B, vol. 17, No. 3, pp. 376-382, Aug. 1994.

[24] W. R. Eisenstadt and Y. Eo, *"S-Parameter-Based IC Interconnect Transmission Line Characterization,"* IEEE Trans. CPMT., vol. 15, No. 4, pp. 483-490, Aug. 1992.

[25] C. C. Courtney, *"Time-Domain Measurement of the Electromagnetic Properties of Materials,"* IEEE Trans. MTT., vol., 46, No. 5, pp. 517-522, May 1998.

[26] J. Wee et al., *"Modeling the Substrate Effect in Interconnect Line Characterisitics of High-Speed VLSI Circuits,"* IEEE Trans. MTT., vol., 46, No. 5, pp. 1436-1443, Oct. 1998.

[27] D. F. Williams, U. Arz, and H. Grabinski, *"Accurate Characteristic Impedance Measurement on Silicon,"* IEEE MTT-S Int. Microwave Symposium Digest, 1998, pp. 1917-1920.

[28] J. E. Schutt-Aine, R. Mittra, *"Scattering Parameter Transient Analysis of Transmission Lines Loaded with Nonliear Terminations,"* IEEE Trans. MTT., vol. 36, No. 3, pp. 529-536, Mar. 1988.

[29] R. Achar, M. Nakhla, and E. Ahmed, *"Nonlinear Transient Simulation of Embedded Subnetworks Characterized by S-parameters Using Complex Frequency Hopping,"* IEEE MTT-S Int. Microwave Symposium Digest, 1998, pp. 267-270.

[30] A. Verschueren et al., *"Identifying S-Parameter Models in the Laplace Domain for High Frequency Multiport Linear Network,"* IEEE MTT-S Int. Microwave Symposium Digest, 1998, pp. 25-28.

[31] P. J. Restle, K. Al Jenkins, A. Deutch, and P. W. Cook, *"Measurement and Modeling of On-Chip Transmission Line Effects in a 400 MHz Microprocessor,"* IEEE JSSC., vol. 33, No. 4, pp. 662-665, Apr. 1998.

[32] D. W. Bailey and B. J. Benschneider, *"Clocking Design and Analysis for a 600-MHz Alpha Microprocessor,"* IEEE JSSC, vol. 33, No. 11, pp. 1627-1633, Nov. 1998.

[33] J. -H. Chern, J. Huang, L. Arledge, P. -C. Li, and P. Yang, *"Multilevel metal capacitance models for CAD design synthesis systems,"* IEEE Electron Device Letters, vol. 13, no. 1, pp. 32-34, Jan. 1992.

[34] J. Cong, L. He, A. B. Kahng, D. Noice, N. Shirali and S. H. -C. Yen, *"Analysis and justification of a simple, practical 2 1/2-D capacitance extraction methodology,"* in Proc. 34th Design Automation Conf., 1997, pp. 627-632.

[35] N. D. Arora, K. V. Raol, R. Schumann, and L. M. Richardson, *"Modeling and extraction of interconnect capacitance for multilayer VLSI circuits,"* IEEE Trans. Computer-Aided Design, vol. 15, no. 1, pp. 58-67, Jan. 1996.

[36] U. Choudhury and A. Sangiovanni-Vincentelli, *"Automatic generation of analytical models for interconnect capacitances,"* IEEE Trans. Computer-Aided Design, vol. 14, no. 4, pp. 470-480, Apr. 1995.

[37] W. T. Weeks, *"Calculation of coefficients of capacitance of multiconductor transmission lines in the presence of a dielectric interface,"* IEEE Trans. Microwave Theory Tech., vol. MTT-18, no. 1, pp. 35-43, Jan. 1970.

[38] A. E. Ruehli and P. A. Brennan, *"Capacitance Models for Intergated Circuit Matalization Wires,"* IEEE Journal of Solid State Circuit, vol. SC-10 No. 6, pp. 530-536, Dec., 1975.

[39] E. Bogatin and Gaga Test Labs, *Microwave Basic's Short Course,* CA, 1992.

[40] P. R. Gray and R. G. Meyer, *Analysis and Design of Analog Intergrated Circuits,* John Wiley & Sons, Inc., NewYork, 1984.

[41] C. Wei, R. H. Harrington, J. R. Mautz, and T. K. Sarkar, *"Multiconductor transmission Lines in Multilayered Dielectric Media,"* IEEE Trans. MTT vol. 32, no. 4, pp. 439-450, Apr., 1984.

[42] J. S. Ko, B. K. Kim, and K. Lee, *"Simple modeling of coplanar waveguide on thick dielectric over lossy substrate,"* IEEE Trans. on EC, vol.44, no.5, pp.856-861, May 1997.

[43] Y. R. Kwon, V. M. Hietala, and K. S. Champlin, *"Quasi-TEM analysis of slow-wave mode propagation on coplanar microstructure MIS transmission lines,"* IEEE Trans. Microwave Theory & Tech., vol.35, no.6, pp.545-551, June 1987.

[44] M. W. Beattie and L. T. Pileggi, *"IC analyses including extracted inductance models,"* Proc. 36th ACM/IEEE Design Automation Conf., pp.915-920, 1999.

[45] S. V. Morton, *"On-chip inductance issues in multiconductor systems,"* Proc. 36th ACM/IEEE Design Automation Conf., pp.921-926, 1999.

[46] J. A. Davis and J. D. Meindl, *"Compact distributed RLC models for multilevel interconnect networks,"* 1999 Symposium on VLSI Circuits Digest of Technical Papers, 1999.

[47] W. Jin, S. Yoon, Y. Eo, and J. Kim, *"Experimental Characterization and Modeling of Transmission Line Effects for High-Speed VLSI Circuit Interconnects,"* IEICE Trans. on Electronics, vol E83-C, no. 5, pp. 728-735, May 2000.

[48] Y. Ikawa, W. R. Eisenstadt, and R. W. Dutton, *"Modeling of High-Speed, Large-Signal Transistor Switching Transients from S-Parameter Measurements,"* IEEE Trans. ED., vol-ED 29, no. 4, pp. 669-675. 1982.

[49] J. L. Wyatt, Jr., *"Signal Delay in RC Mesh Networks,"* IEEE Trans. CAS, vol. CAS-32, pp. 507-510, May 1985.

[50] M. Hatamian, L. A. Hornak, E. E. Little, S. K. Tewksbury, and P. Franzon, *"Fundamental Interconnect Issues,"* AT&T Technical Journal, vol. 66, Issue 4, pp. 13-30, Jul. 1987.

[51] A. N. Saxena, *"Interconnect for the '90s: Alumimum-Based Multilevel Interconnects and Future Directions,"* IEDM 1992 Short Course: Interconnect for the '90s, San Jose, CA, 1992.

[52] A. R. Djordjevic, T. K. Sarkar, and R. F. Harrington, *"Time Domain Response of Multiconductor Transmission Lines,"* Proceedings of IEEE, vol. 75, No. 6, pp. 743-764, Jun. 1987.

[53] G. Ghione, I. Maio, and G. Vecchi, *"Modeling of Multiconductor Buses and Analysis of Crosstalk, Propagation Delay and Pulse Distortion in High-Speed GaAs Logic Circuits,"* IEEE Trans. MTT., vol. 37, No. 3, pp. 445-456, Mar. 1989.

[54] D. Winklestein, M. B. Steer and R. Pomerleau, *"Simulation of Arbitrary Transmission Line Networks with Nonlinear Terminations,"* IEEE Trans. CAS., vol. 38, No. 4, pp. 418-422, Apr. 1991.

[55] H. You and M. Soma, *"Crosstalk Analysis of Interconnect Lines and Packages in High Speed Integrated Circuits,"* IEEE Trans. CAS., vol. 37, No. 8, pp. 1019-1026, Aug. 1990.

[56] D. Nayak, et al., *"Calculation of Electrical Prameters of a Thin-Film Multichip Package,"* IEEE Trans. CHMT., vol. 12, No. 2, pp. 303-369, Jun. 1989.

[57] J. C. Isaacs, Jr. and N. A. Strakhov, *"Crosstalk in Uniformly Coupled Lossy Transmission Line,"* Bell Syst. Tech. J., vol. 52, No.1, pp. 101-111, Jan. 1973.

[58] M. Shoji, *CMOS Digital Circuit Technology,* Englewood Cliffs, NJ: Prentice- Hall, 1988.

[59] F. Dartu and L. T. Pileggi, *"Calculating Worst-Case Gate Delays Due to Dominant Capacitance Coupling,"* in Proc. 34th Design Automation Conf., pp. 46-51, Jun. 1997.

[60] C. P. Yuan and T. N. Trick, *"A Simple Formula for the Estimation of the Capacitance of Two-Dimensional Interconnects in VLSI Circuits,"* IEEE Electron Device Letter, vol. EDL-3, No. 12, pp. 391-393, Dec. 1982.

[61] R. Evans and M. Tsuk, *"Modeling and Measurement of a High-Performance Computer Power Distribution System,"* IEEE Trans. CPMT-Part B. vol. 17, No. 4, pp. 467-471, Nov. 1994.

[62] A. Vaidyanath, B. Thoroddsen, and J. L. Prince, *"Effect of CMOS Driver Loading Conditions on Simultaneous Switching Noise,"* IEEE Trans. CPMT, vol. 17, pp. 1724-1728, Nov. 1991.

[63] S. R. Vemuru, *"Accurate Simultaneous Switching Noise Estimation Including Velocity-Saturation Effect,"* IEEE Trans. CPMT-PAPT B, vol., 19, No.2, pp. 344-349. May 1996.

[64] R. Senthinathan et al., *"Electrical Package Requirements for Low-Voltage Ics-3.3V High-Performance CMOS Devices as a Case Study,"* IEEE Trans. CPMT-Part B, vol. 17, No. 4, pp. 493-503, Nov. 1994.

[65] A. J. Rainal, *"Computing Inductive Noise of CMOS Drivers,"* IEEE Trans. CPMT.- Part B, vol. 19, No.4, pp. 789-802, Nov. 1996.

[66] K. Bathey, M. Swaminathan, L. D. Smith, and T. J. Cockerill, *"Noise Computation in Single Chip Packages,"* IEEE Trans. CPMT-Part B, vol. 19, No. 2, pp. 350-360, May 1996.

[67] J.-G. Yook et al., *"Computation of Switching Noise in Printed Circuit Boards,"* IEEE Trans. CPMT.-Part A, vol. 20, No. 1, pp. 64-75, Mar. 1997.

[68] T. J. Gabara et al, *"Forming Damped LRC Parasitic Circuits in Simultaneously Switched CMOS Output Buffers,"* IEEE JSSC, vol. 32, No. 3, pp. 407-418, Mar. 1997.

[69] H. I. Hanafi et al., *"Design and Characterization of a CMOS Off-Chip Driver/Receiver with Reduced Power-Supply Disturbance,"* IEEE JSSC., vol. 27, No. 5, pp. 783-791, May 1992.

[70] R. Senthinathan and J. L. Prince, *"Application Specific CMOS Output Driver Circuit Design Techniques to Reduce Simultaneous Switching Noise,"* IEEE JSSC. vol. 28, No. 12, pp. 1383-1388, Dec. 1993.

[71] Y. Yang and J. R. Brews, *"Design Trade-Offs for the Last Stage of an Unregulated, Long-Channel CMOS Off-Chip Driver with Simultaneous Switching Noise and Switching Time Consideration,"* IEEE Trans. CPMT-PART B, vol. 19, No. 3, pp. 481-486, Aug. 1996.

[72] Y. Yang and J. R. Brews, *"Design for Velocity Saturated, Short-Channel CMOS Drivers with Simultaneous Switching Noise and Switching Time Considerations,"* IEEE JSSC. vol., 31 No. 9, pp. 1357-1360, Sep. 1996.

[73] D. A. Secker and J. L. Prince, *"Effects and Modeling of Simultaneous Switching Noise for BiCMOS OFF-Chip Drivers,"* IEEE CPMT.-Part B, vol., 19, No. 3 pp. 473-480, Aug. 1996.

[74] Y. Yang and J. R. Brews, *"Guidelines for High-Performance Electronic Package Interconnections-Approach for Strong Coupling,"* IEEE Trans. CPMT.-Part B., vol. 19, No. 2, pp. 372-381, May 1996.

[75] T. Sakurai and A. Newton, *"Alpha-Power Law MOSFET Model and Its Application to CMOS Inverter Delay and Other Formulas,"* IEEE JSSC, vol. 25, pp. 584-549, Apr. 1990.

[76] M. E. Van Valkenburg, *Network Analysis,* Reading, NY: Prentice Hall, 1974, Ch.6, pp.139-163.

[77] P. Larsson, *"Resonance and Damping in CMOS Circuits with On-Chip Decoupling Capacitance,"* IEEE Trans. Circuits and System-I: Fundamental Theory and Applications, vol. 45, No. 8, pp. 849-858, Aug. 1988.

[78] Y. Eo, W. R. Eisenstadt, J. Y. Jeong, and O-. K. Kwon, *"A New On-Chip Interconnect Crosstalk Model and Experimental Verification for CMOS VLSI Circuit Design,"* IEEE Trans. on Electron Devices, vol. 47, no. 1, pp. 129-140, Jan. 2000.

[79] Y. Eo, W. R. Eisenstadt, J. Y. Jeong, and O-. K. Kwon, *"New Simultaneous Switching Noise Analysis and Modeling for High-Speed and High-Density CMOS IC Package Design,"* IEEE Trans. on Advanced Packaging, vol. 23, no. 2, pp. 303-312, May 2000.

[80] Y. Eo, W. R. Eisenstadt, and J. Shim, *"S-Parameter-Measurement-Based High-Speed Signal Transient Characterization of VLSI Interconnects,"* IEEE Trans. on Advanced Packaging, vol. 23, no. 3, pp. 470-479, Aug. 2000.

[81] *"MEDICI User's Manual,"* TMA Corp., Sunnyvale, CA.

Tradeoffs in Digital Binary Adder Design: the effects of floorplanning, number of levels of metals, and supply voltage on performance and area

Vitit Kantabutra
College of Engineering, Box 8060
Idaho State University, Pocatello, ID 83209, U.S.A.
E-mail: vkantabu@computer.org

Stefania Perri
Department of Electronics, Computer Science and Systems
University of Calabria, Arcavacata di Rende, 87036 Rende(CS), Italy
E-mail: perri@deis.unical.it

Pasquale Corsonello
Department of Informatics, Mathematics, Electronics, and Transportation
University of Reggio Calabria, Loc. Vito de Feo, 89060 Reggio Calabria, Italy
E-mail: pascor@deis.unical.it

Contents

B. Lu et al. (eds.), Layout Optimization in VLSI Design, 261–288.
© 2001 *Kluwer Academic Publishers.*

References

1 Introduction

Arithmetic circuits play a crucial role in most complex digital systems today. Virtually all complex digital systems contain arithmetic circuits in their critical paths, and thus the performance of arithmetic circuits can greatly affect the performance of the system as a whole. Likewise the area and power dissipation of arithmetic circuits can be important factors in the determination of the feasibility of the system. Area and power issues appear to be even more important today because circuits are often meant to perform in less than optimum conditions, such as where components are tightly packed and don't necessarily have large amounts of ventilation. Thus the realization of area- and time-efficient arithmetic circuits is of fundamental importance. This is especially true for adders, which appear in all arithmetic circuits. Unfortunately, establishing what kind of adder is the most appropriate for a specific application is not trivial. In fact, in order to achieve the best area-time trade-off, the designer should analyze the characteristics of several adders. In addition, sometimes the designer is given a fixed portion of the chip area for the circuit, so that there is no flexibility in the shape that the circuit can take. This implies that dependencies of performance and cost on topology and layout constraints should be both taken into account.

Two methods are available for producing layout: automatic and manual. Additionally a mixed approach is possible, where only part of the layout is automated. Manual place and route allows detailed changes to be done to the layout. This can be useful for reducing parasitics and for improving the placement of gates leading to higher performance and lower area requirement. However, to minimize the time for testing and realization fully automated approaches are usually preferred. Of these automated approaches the most widespread one is the standard-cells place and route process involving only the standard cells libraries provided by the foundries. Note that the place and route software supporting this approach allows several parameters to be set, influencing area and performance of the circuit to be laid out [12]. For example, the designer is able to choose the layout aspect ratio (sometimes indirectly by means of choosing the number of rows of cells) and the number of levels of metal and polysilicon to be used during the routing process. Obviously, appropriate values have to be chosen for these parameters taking into account design specifications and constraints.

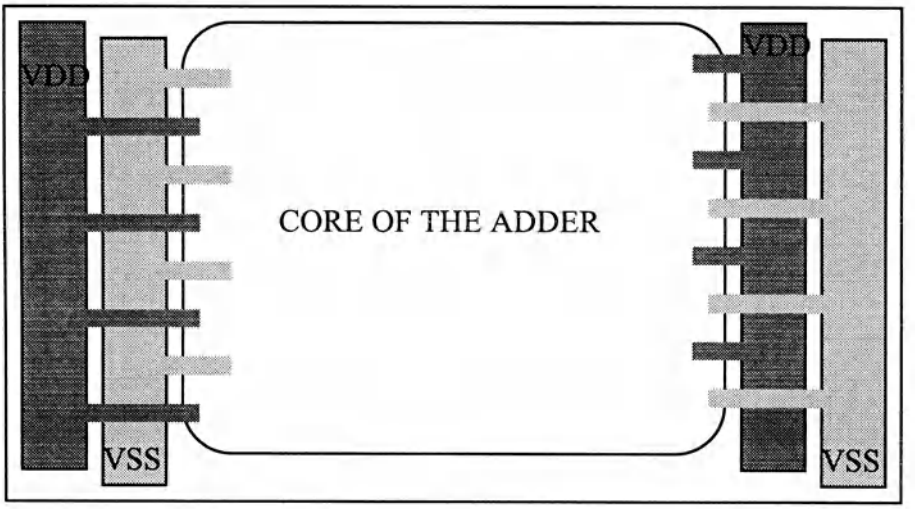


Figure 1: Lateral power supply lines are very large

It is well known that the number of rows of cells in a layout or the aspect ratio can affect Circuit Delay and Area for the following reasons:

1. Usually, the main ("lateral") power supply lines that run on the left and right sides of a circuit (Figure. 1) are very large. This is necessary to supply current to all the branches of the circuit avoiding metal migration [12]. The active area of the circuit does not change when the aspect ratio varies but a higher number of rows of cells leads to a higher area for the main supply lines. The power supply area change can be quite significant.

2. Gate-to-gate as well as port-to-gate (I/O) interconnection lengths change if the number of rows of cells varies. With few rows of cells gate-to-gate interconnections can be critical because long connections may be necessary simply to reach from one place to another. This is the case shown, for example, in Figure. 2. As the number of rows increases to an intermediate value (e.g. 4), the I/O interconnections can become critical. In fact, as shown in Figure. 3 they may reach half the layout width. Once the number of rows becomes very high, gate-to-gate interconnections dominate the I/O ones once more. As shown in Figure. 4 this happens because no cell is very far from the chip edge, and because some logic signals may have to travel between two distant rows of cells. From the discussion above it is clear that changes to

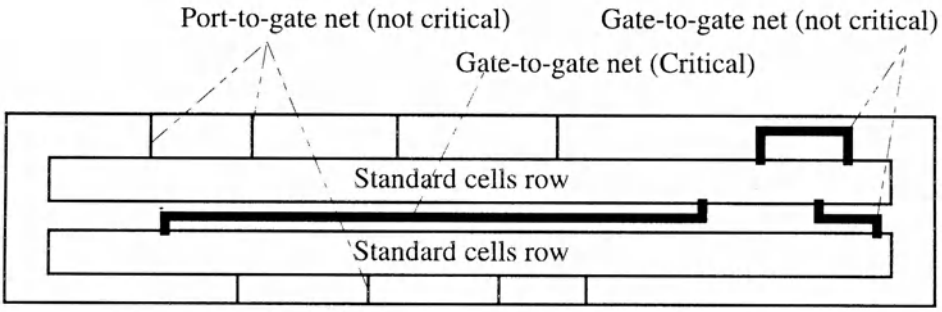


Figure 2: With few rows of cells, long interconnections are often needed

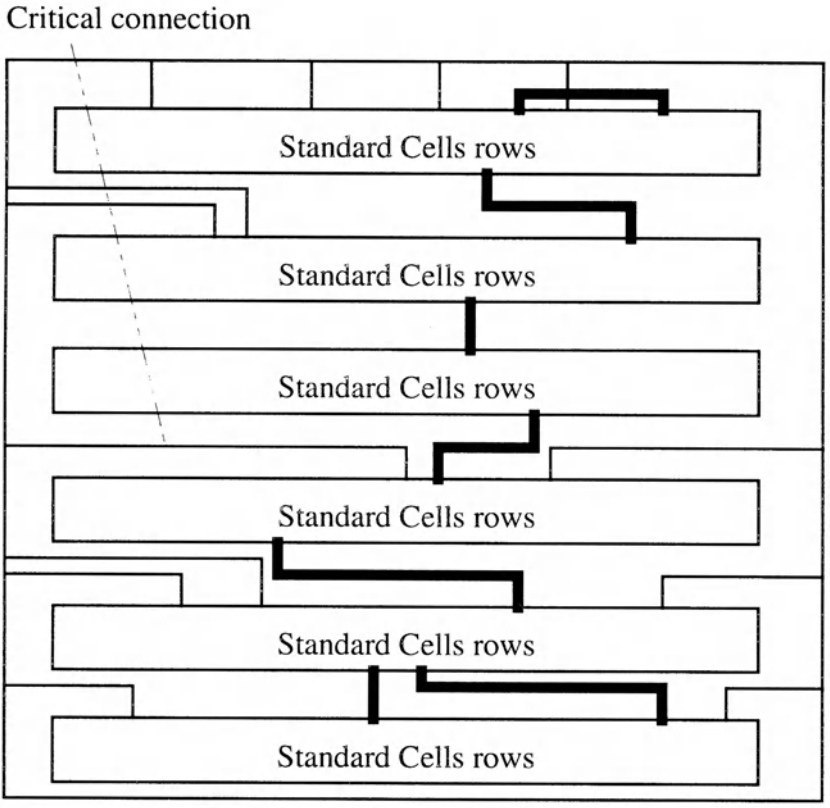


Figure 3: I/O interconnection lengths may reach half the layout width

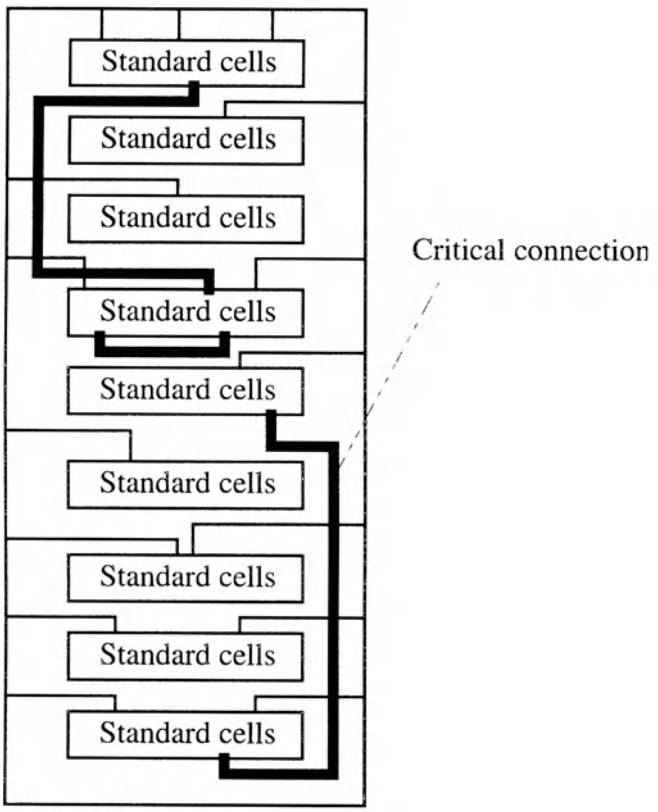


Figure 4: As the number of row rises gate-to-gate interconnections can dominate I/O interconnections once more

the layout aspect ratio can significantly influence net capacitance and consequently net delays.

As it is well known, the number of levels of metal also influences circuit delay and area. Today technologies with up to 10 levels of metal are available. In particular a higher number of levels of metal is expected to facilitate routing for the following reasons:

1. The first obvious advantage is the reduction on the wiring congestion, which leads to smaller routing channels. This implies a reduction on the silicon area requirement.

2. Sometimes, using few levels of metal, routing complex nets or a large number of nets in a small area can be difficult. In such situations winding paths may be necessary for critical nets and sometimes stretches of poly may be required to resolve connections that are not routable with metal alone. Increasing the number of levels of metal avoids complex paths and stretches of high-capacitance poly. Obviously this leads to a gain in speed.

Obviously supply voltage changes do not affect area requirement, but they influence power dissipation and delay.

This chapter presents three separate but related sets of experimental results that pertain to the effects of the above parameters on performance and area requirement of binary adders realized using the fully automated standard cells approach. The 32-bit adders we will refer to are the following: a Ripple-Carry, a conventional Carry-Skip, a Carry-Select, a Carry Lookahead, and the recently proposed Carry-Strength Adder [4]. The sets of results we present in this chapter will show:

- The effects of floorplanning on area and delay,
- The effects of the number of levels of metal on area and delay, and,
- The effects of supply voltage on delay.

Of these three sets of results, we will devote most of our effort on the first one, namely, the effects of floorplanning on area and delay of binary adders implemented in a standard cell technology. The technology we used is AMS (Austria Mikro Systeme) $0.6\mu m$ standard cell (Hit-Kit version 3.20), running under the IC Station tool V8.7_2.1 of the Mentor Graphics design system.

Floorplanning is normally defined as the placement of blocks of fixed area but unknown dimension of shape [10]. However, due to the large size

of power supply lines, we will find that the layout area of a circuit with fixed active area can vary quite significantly as we change the shape of the layout. We will also find that the worst-case delay of a circuit can change according to the floorplan, and we will study the cause of this change. In particular, we will find that the layout area can, in the worst case, almost double as the number of rows of cells increases from 2 to 14. Circuit delay also suffers significant changes with respect to the number of layout rows. We will find that the changes in the delay are smaller and more complex than the changes in area. Note, however, that in larger circuits the delay changes may be more pronounced due to the possibility of longer wires.

The results of this study emphasize the importance of careful global floorplanning, where the aspect ratio of each block is allowed to vary. Our results also highlight the need for new floorplanning algorithms, where constraints on performance and area of a block should be input to the algorithms. Floorplanning algorithms that take circuit performance seriously into account appear to be half a decade old [2, 8, 9, 12, 13].

The analysis of the effects of the layout aspect ratio on circuit delay and area will allow the demonstration of the superior properties of the Carry-Strength adder in terms of silicon area requirement and performance. Due to this, we investigated on the effects of the number of levels of metal on carry-strength delay and area. In particular, our second set of results quantifies for the carry-strength adder what happens when we increase the number of levels of metal from 2 to 3. There we will find that the area decreases by 14% on average for the number of rows studied (2, 3, 4, 5, 6, 10, and 14), whereas the delay decreases only slightly, bringing the area-delay product decreases by 15% on average.

The last set of results involves checking the delays of all the above-mentioned adders at different voltages between 2 and 5 volts. There we find that the carry-strength adder is the most robust and reliable addition circuit, since it is much less sensitive to the supply voltage changes than the other adders.

Section 2 is devoted to the description of the adders involved in our experimental analysis, whose results will be presented in sections 3 and 4.

2 Binary Adders

The importance of a fast, low-cost binary adder in a digital system is difficult to overestimate. Not only are adders used in every arithmetic operation, they are also needed for computing the physical address in virtually every

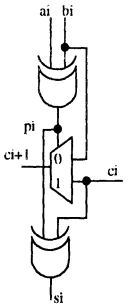


Figure 5: Typical one-bit full adder

memory fetch operation in most modern CPUs. Adders are also used in many other digital systems including telecommunications systems in places where a full-fledged CPU would be superfluous.

Many styles of adders exist, each with its own area-delay trade-off. Ripple-carry adders are known as the smallest adders but are also the slowest. Better performance with reasonable silicon area increase can be achieved with the carry-skip adders [4, 7]. Carry-select and carry lookahead adders [4, 7] are very fast but far larger than ripple-carry or traditional carry-skip adders. Recently, a new kind of carry-skip adder, here named carry-strength adder, has been proposed. It offers the advantage of performance comparable with the carry lookahead adder while maintaining a silicon area requirement not much greater than that of the conventional carry-skip adder. In the following, all the above mentioned adders will be described, but first of all their hardware addition algorithms will be studied.

The basic adder is known as a one-bit full adder, which computes a sum bit and a carry-out from two one-bit addends and a carry-in. The full adder's functions are as follows:

$$s_i = a_i \oplus b_i \oplus c_i \quad (1)$$

$$c_{i+1} = a_i b_i + b_i c_i + c_i a_i \quad (2)$$

where s_i and c_{i+1} are the sum bit and the carry-out, respectively, whereas a_i, b_i and c_i are the addends and the carry-in, respectively. Figure. 5 shows a typical gate-level implementation of a one-bit full adder. The n-bit ripple-carry adder is built from n one-bit full adders. This is shown in Figure. 6, where the 32-bit ripple-carry adder is illustrated. Note that the sum bit and the carry-out generated by the ith full adder depend on the carry-out generated by the (i-1)th full adder. This implies that the critical delay path of the ripple-carry adder goes from the 0-bit inputs to the n-1 bit up through

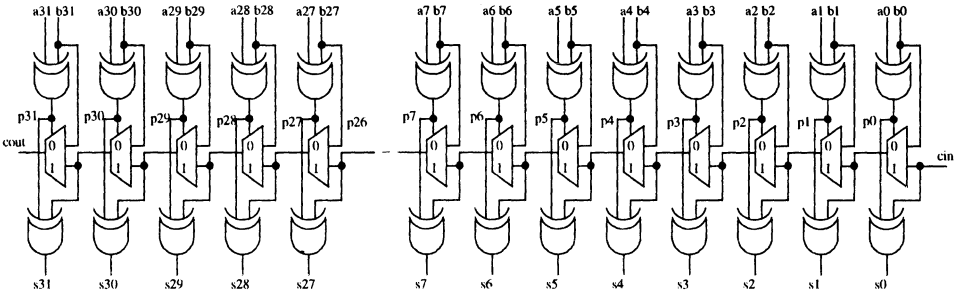


Figure 6: A 32-bit ripple adder

the carry chain. The ripple-carry adder is area-efficient and easy to design but it is slow for any moderate or large values of n. It is easy to understand that for the case shown in Figure. 6 the critical path delay consists of 31 multiplexers and two EXOR gates for the sum computation, whereas it consists of 32 multiplexers and an EXOR gate for the carry output. Several methods exist to speed up the carry chain. An example is the principle used in the carry-select adder. A typical structure of this adder is shown in Figure. 7. In a carry-select adder m-bit stages are used, each consisting of m one-bit full adders (Figure. 7). All stages, the first one excepted, compute two versions of the sum with the possible carry-ins of 0 and 1. When the true carry-in is ready the correct sum and carry output bits are selected by multiplexers controlled by the carry output of the previous stage. This method allows speeding up the carry chain since the two versions of additions are computed in parallel. Thus, the delay of each stage subsequent to the first one is limited to that of a 2:1 multiplexer. For the 32-bit version of the carry-select adder this leads to the critical path consisting of 11 multiplexers and an EXOR gate for both the sum and carry-out computations.

Another useful method for speeding up addition is that which is used in the carry lookahead adder. The carry-lookahead adder operates according to the idea that the carry into each bit position is a Boolean function of the bit positions to the right, that is, the less significant bits. Therefore it is possible (but costly) to compute this bit carry-in in a more-or-less direct fashion instead of letting the carry ripple through the less significant bits. The carry-lookahead adder performs its computation into two steps:

1. First compute the Propagate (P) and Generate (G) signals as shown

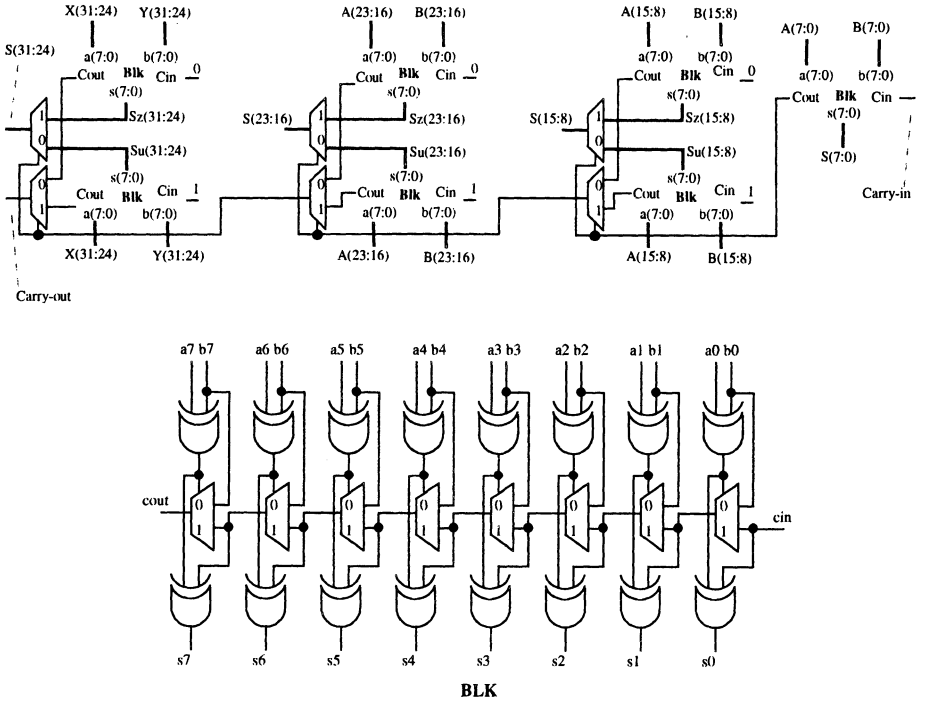


Figure 7: Typical structure of a 32-bit carry-select adder

in equations 3 and 4::

$$P_i = a_i \oplus b_i \tag{3}$$
$$G_i = a_i \cdot b_i \tag{4}$$

2. Then obtain the sum and carry outputs by means of functions of P_i and G_i as shown in equations 5 and 6:

$$s_i = c_i \oplus P_i \tag{5}$$
$$c_{i+i} = G_i + P_i \cdot c_i \tag{6}$$

The carry formula can be recursively expanded, then it can be written as in equation 7:

$$\begin{aligned} c_{i+1} &= G_i + P_i \cdot (G_{i-1} + P_{i-1} \cdot c_{i-1}) \\ &= G_i + P_i \cdot G_{i-1} + P_i \cdot P_{i-1} \cdot (G_{i-2} + P_{i-2} \cdot c_{i-2}) \\ &= G_i + P_i \cdot G_{i-1} + P_i \cdot P_{i-1} \cdot G_{i-2} + \\ &\quad P_i \cdot P_{i-1} \cdot P_{i-2} \cdot (G_{i-3} + P_{i-3} \cdot c_{i-3}) \\ &= \sum_{k=0}^{i} \left(G_k \cdot \prod_{j=k+1}^{i} P_j \right) \end{aligned} \tag{7}$$

Note that the carry c_{i+1} is expressed in terms of P, G, and the carry-in of the adder (c_0). This implies a very short carry chain. However, it is easy to understand that there is a limit beyond which larger gates with higher fan-in are necessary, reducing the gain in speed obtainable with a carry lookahead adder. Typically, four levels of carry can be usefully expanded without compromising speed up. Taking this into account the 32-bit version of the carry lookahead adder can be structured as shown in Figure. 8. Observing the above circuits it can be easily verified that the critical delay path consists of $3 \cdot And_4 + 3 \cdot Or_4 + And_2 + Or_2 + 2 \cdot Xor_2$ for the sum computation, whereas it consists of $2 \cdot And_4 + 2 \cdot Or_4 + And_3 + Or_3 + Xor_2$ for the carry-out generation.

The carry-skip adder uses the carry-propagate and carry-generate relationships 3 and 4 differently. In fact, it looks for cases in which the carry-out of a set of bits is the same as the carry-in into those bits. Usually, a carry-skip adder is divided into a proper number of m-bit blocks, each representing a set of m bits [4, 7]. In Figure. 9 is illustrated the 32-bit version of a carry-skip adder using 8-bit blocks. Each non-least significant 8-bit block can be structured as shown in Figure. 10.

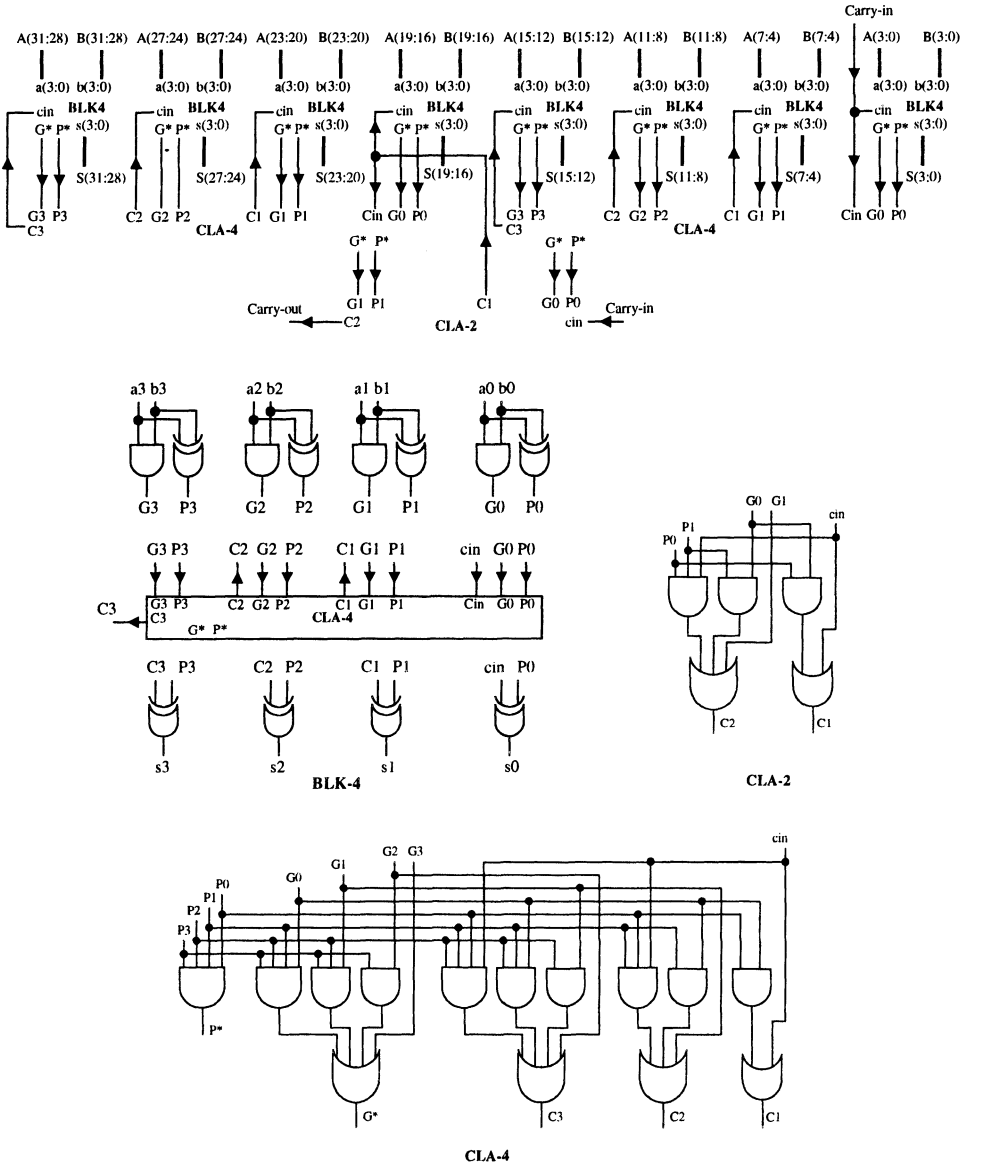


Figure 8: Typical structure of a 32-bit carry-lookahead adder

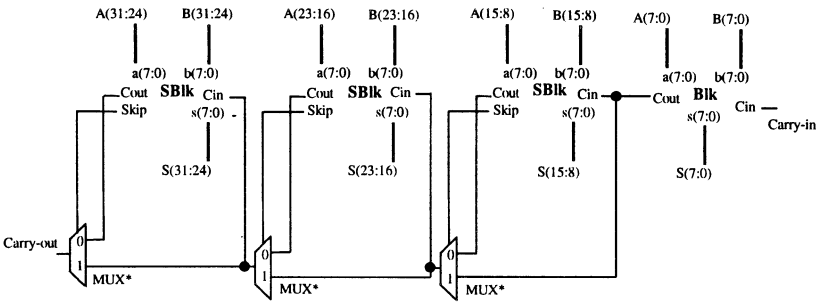


Figure 9: A 32-bit carry-skip adder using 8-bit blocks

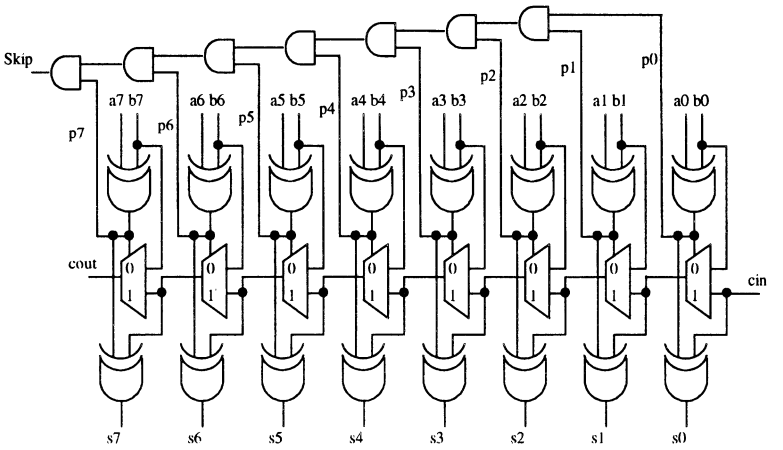


Figure 10: Structure of the generic non-least significant 8-bit block

In the least significant 8-bit block just the AND gates generating the skip signal are not necessary. So that it is structured as shown in Figure. 7(b).

On the basis of the propagate signals $p_7, \ldots, p_0$, each block is able to establish if its carry-out depends or not on the carry-in. If skip=1 the carry-in of the block is "immediately" transferred to the carry-out line by means of the external 2:1 multiplexer MUX*. The skip signal assumes the low logic level only if a carry is internally generated into the block. In fact, when this event occurs one or more of the propagate signals are low. It is worth emphasizing that all the blocks in which skip=0 work in parallel. Thus, a long-range carry signal starts at a generic block B_i, ripples through some bits in that block, then skips some blocks, and ends in a block B_j. If the carry does not end at the LSB of B_j then rippling occurs in that block and an additional delay is needed to compute the valid sum bits. It can be easily verified that for the 32-bit carry-skip adder shown in Figure. 9 the critical delay path consists of 17 multiplexers and 2 EXOR gates for the sum computation, whereas it consists of 18 multiplexers and an EXOR gate for the carry-out validation. The carry chain of a carry-skip adder can be further sped up using the optimization criterion detailed in [3], which suggests the usage of non-uniformly sized block. In that paper it was shown that the block sizes should be small near the MSB and LSB, and gradually larger towards the middle of the bit positions.

Recently, a family of new carry-skip adders, here named carry-strength adders, that are significantly faster than traditional carry-skip adders while not much larger has been proposed [4]. A carry-strength adder is based on the elimination of the delay due to the rippling occurring when a long-range carry dies into a block. In order to this, a carry-strength (CS) signal is defined for each bit position into any non-least significant m-bit block as follows:

$$CS_{k+1} = \begin{cases} \overline{x_k \oplus y_k} & \text{if } k \text{ the LSB of the block} \\ CS_k + (\overline{x_k \oplus y_k}) & \text{otherwise} \end{cases} \quad (8)$$

In other words, for bit position k that is not the LSB of a block of bits, the incoming carry-strength CS_k is high if and only if the carry into the same position (C_k) is independent of the carry-in of the block containing that bit position. In case $CS_k = 1$, we also say that the carry-in C_k is strong. Otherwise C_k is weak. Carry-strength signals are useful in a block in which a long-range carry signal ends. To demonstrate the usefulness of carry-strength signals, let us consider the following two complementary cases. If $CS_k = 0$, that is, the carry is weak, it is easy to verify that C_k corresponds to the block carry-in. Thus, we can just select C_k to be the same as the

block carry-in, eliminating the delay due to rippling. On the other hand, if $CS_k = 1$, then C_k is independent of the carry-in, and is therefore known quickly. (In other words the computation of C_k starts as soon as the adder's operands appear, without waiting for an incoming block carry-in). For these reasons, the existence of carry-strength signals trivializes the delay in the ending block of a long-range carry signal, provided that the block carry-in is fed into a large enough buffer. Carry-strength signals can be easily computed in a ripple fashion implementing the above recursive definition. However, the latter rippling starts right when the operands are ready, not having to wait for the carry-in signal. In the 32-bit version of the carry-strength adder consisting of 8-bit blocks the structure shown in Figure. 11 is used for the non-least significant 8-bit block. The least significant 8-bit block is organized as above shown in Figure. 7b. It can be easily verified that the critical path of such carry-strength adder involves 11 multiplexers and 2 EXOR gates for the sum computation, whereas it consists of 11 multiplexers and an EXOR gate for the carry-out validation. Also the carry chain of the carry-strength adder can be further speed up using differently sized block. For its 32-bit version the optimum architecture is that shown in Figure. 12. Note that the critical delay path consists of 9 multiplexers and 2 EXOR gates for the sum computation, whereas it consists of 9 multiplexers and an EXOR gate for the carry-out validation. All the above-described 32-bit adders have been realized and their characteristics have been analyzed highlighting the effects of layout aspect ratio and of the number of levels of metal on circuit delay and area. The obtained experimental results are discussed in the following section.

3 The Effects of the Layout Aspect Ratio and of the Number of Levels of Metal on Circuit Delay and Area

In order to analyze the effects of the layout aspect ratio and of the number of levels of metal on circuit delay and area, we automatically laid out all the above adders. For all these layouts, we used the AMS (Austria Mikro Systeme) $0.6\mu m$ standard cell technology and the Mentor Graphics design system. In a first analysis 2 levels of metal (CUB process) were used. Then the effects of adding a third layer of metal (CUD process) on the overall performance of the carry-strength adder was also studied for the case in which all adders have uniformly sized blocks (i.e. 8-bit blocks).

For all the adders described in section 2 several layouts have been pro-

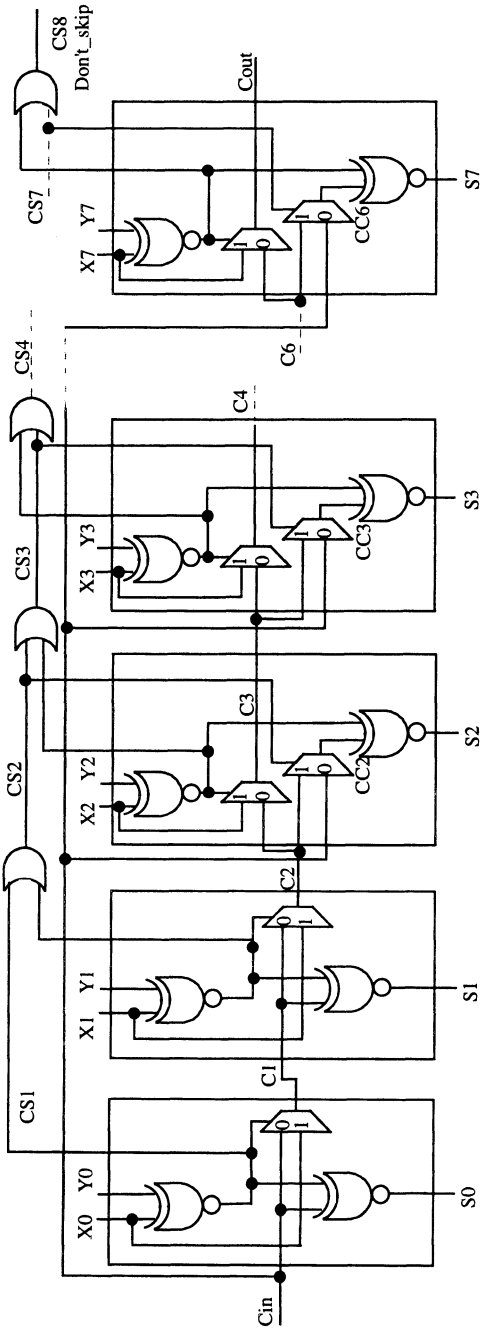


Figure 11: Structure of the non-least significant 8-bit block used in the carry-strength ader.

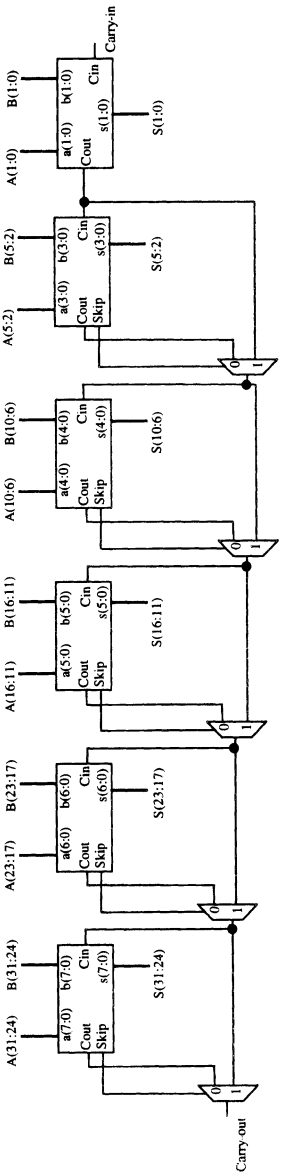


Figure 12: Optimum 32-bit adder with carry strength

duced. Instead of varying the aspect ratio directly as a parameter in our experiments, we produced all layouts varying the number of rows of cells that is an input expected by the Mentor place and route software. In Figure. 13 the total layout area (a), lateral power routing area (b), and active core area (c) are reported for all the studied adders and for several numbers of rows. It can be seen that, as expected, the layout area increases as the number of rows grows and that this is mainly due to the greater lateral power routing area.

In Figure 14 the worst-case delays of the adders we have studied are shown, each one normalized to the minimum value measured over all the numbers of rows of cells. It can be noted that not only the delay depends on the layout aspect ratio as expected, but also that some adders are more sensitive to the changes in the aspect ratio. This is due to the difference in the topology of the different types of adders.

Among the adders we analysed the carry lookahead seems to be the most sensitive one in terms of worst-case delay.

All the above data have been obtained using automated layout with simulated annealing and two levels of compaction. Many tests have been done to try to obtain better results by means of manual pre-placement. However, these attempts have been successful only for the ripple-carry adder. Even in this case, the area saving is very small with respect to the automated layouts and it is limited to the case in which the floorplan is organized into few rows (1, 2, 3, and 4). For all the other adders, manual pre-placement has given worse results than automatic place and route.

In order to observe how net capacitance depends on the layout aspect ratio we evaluated I/O and gate-to-gate net capacitance. Figure. 15 shows the ratio between the capacitance of the longest I/O net and the capacitance of the longest gate-to-gate net, for all the adders studied here. The experimental results show that the I/O net dominates until a certain number of rows, then the gate-to-gate net becomes dominant. Note that in the carry lookahead adder, for any of the number of rows studied, the longest I/O net is that which is associated with the carry-in (cin). This is obvious since cin has to feed several modules. We should emphasize that only in the carry lookahead and ripple-carry adders the I/O net becomes quite dominant for some numbers of rows. In fact, for other adders the ratio between the capacitance of the longest I/O net and the capacitance of the longest gate-to-gate net is always less than 1. This is due to the topology of carry-select, carry-skip and carry-strength adders. In fact, as shown in Figure. 7 and Figure. 9 a "bridge" is always necessary to transfer the carry-out of a block to the multiplexer generating the carry-out of the subsequent block.

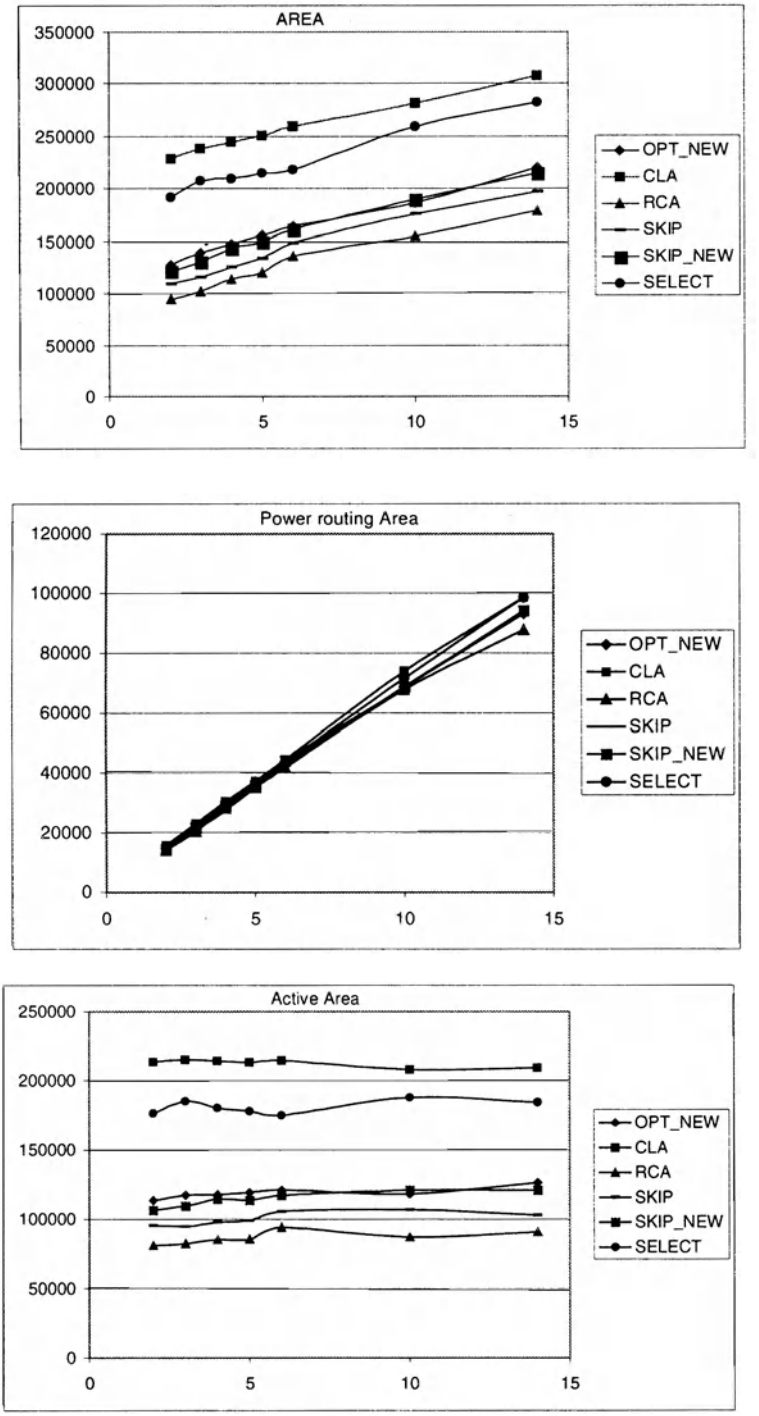


Figure 13: *Top:* (a) Total layout area, *bottom:* (b) Lateral power routing area, (c): Active core area

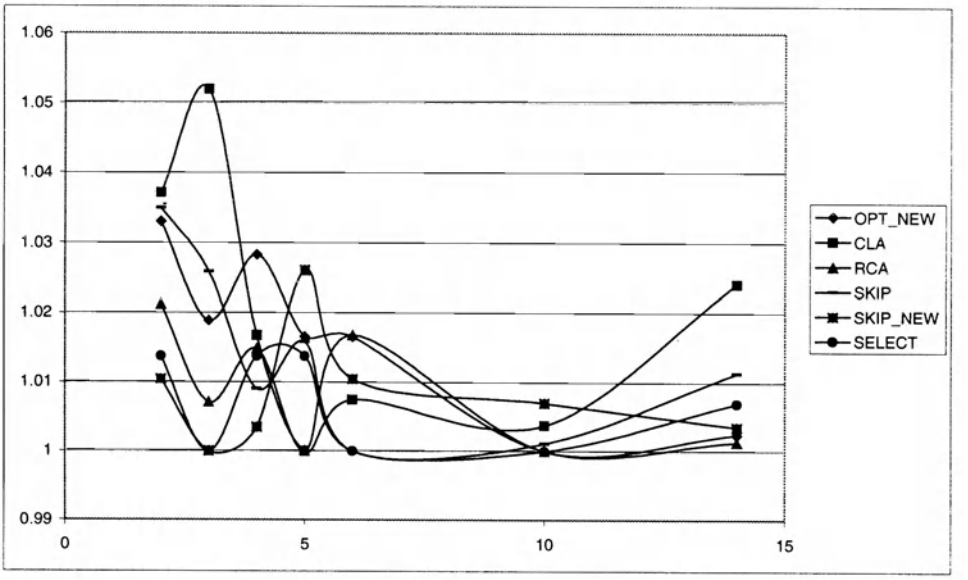


Figure 14: Normalized area

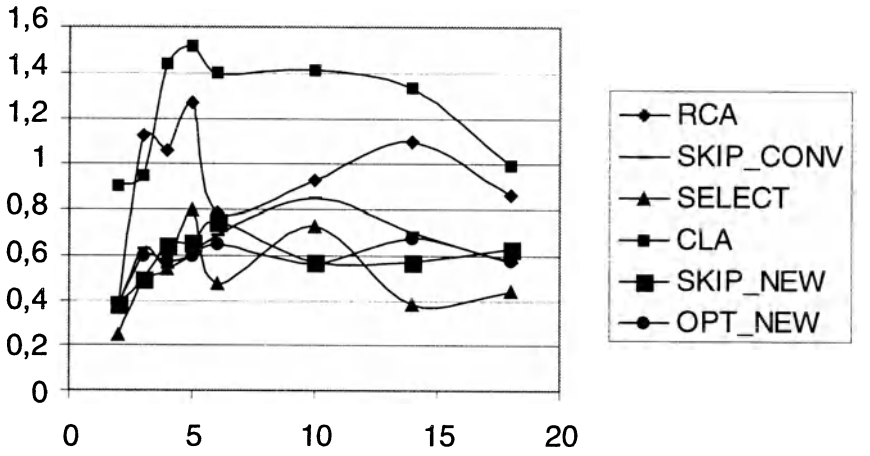


Figure 15: $C_{\text{longest I/O net}}/C_{\text{longest gate-to-gate net}}$

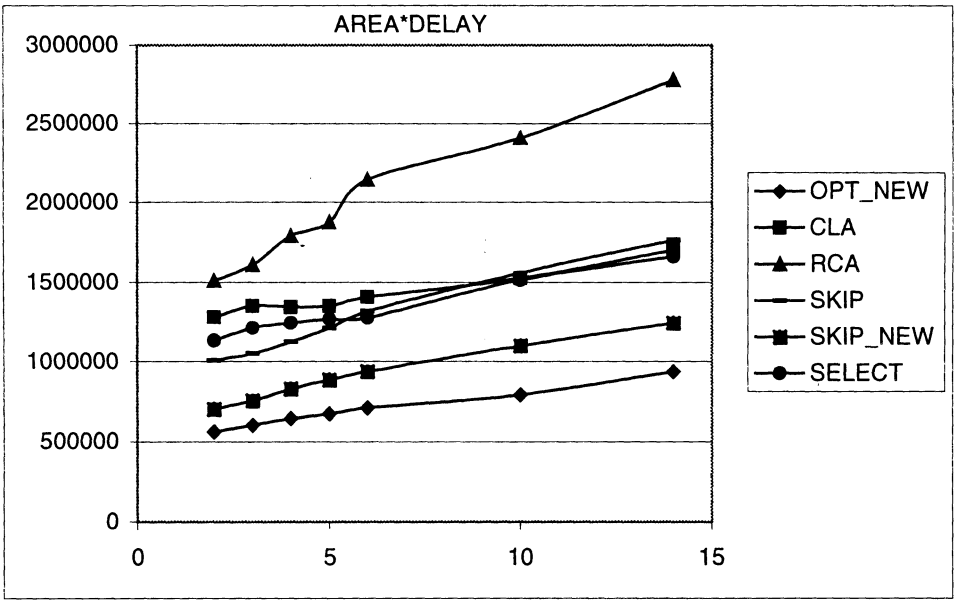


Figure 16: Area-delay product versus aspect ratio

The "bridges" are internal net having intermediate lengths so that the I/O net does not really dominate for any number of rows.

The diagram reported in Figure. 16 serves as good summary of the obtained results. In fact, it shows how the area-delay product depends on the aspect ratio. It can be seen that the carry-strength adders exhibit better characteristics than all the conventional adders. To quantify the advantage of the carry-strength adder without optimized block sizes it has been directly compared to the carry-skip adder with the same block size (8 bits), which shows on the average the best area-delay product among the conventional adders. It is worth emphasizing that the non-optimized carry-strength adder has been referred to for the sake of comparison. In fact the optimized version should be compared to the optimized carry-skip adder. Figure. 17 shows the difference between the area-delay product of the conventional carry-skip adder and that of the carry-strength one. The diagram we obtained shows a steady increase in that difference when the number of rows in the layouts exceeds 5.

The experimental results presented above have been obtained by analyzing layouts in which only 2 levels of metal are used. The benefits of a third level of metal have been quantified for the non-optimized carry-strength

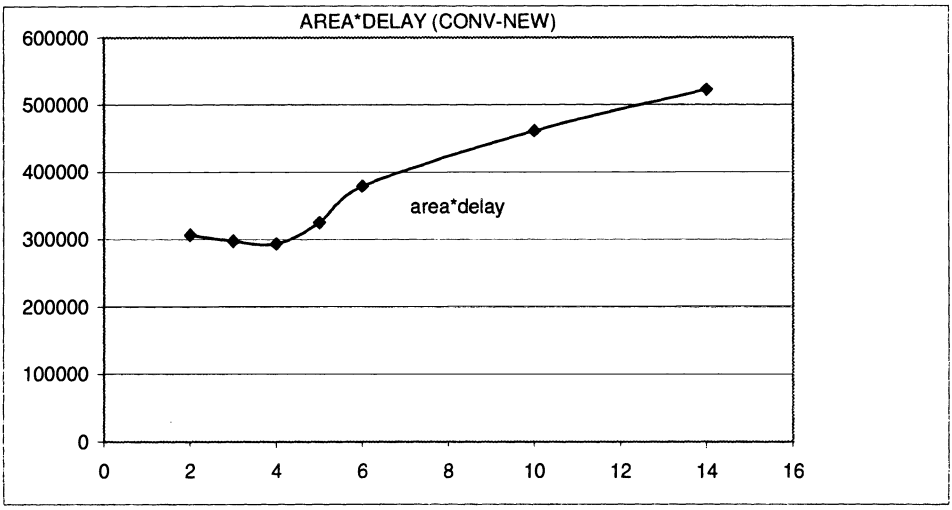


Figure 17: Area-delay product of conventional adders minus area-delay product of new adders

adder. Also in this case several numbers of rows have been tested.

Figure. 18a shows the savings in silicon area due to the use of a third level of metal. As observable in Figure. 18b and Figure. 18c the reduction on the total layout area is due either to the lateral power routing area or to the active core area. In particular, it can be seen that the reduction on the lateral power routing area increases with the number of rows.

Figure 19 shows that the reduction in the circuit delay varies as the number of rows changes and becomes more pronounced as the number of rows becomes very high. This is due to the fact that in this case a third level of metal is essential in making the critical paths shorter than that which could have been routed using only 2 levels of metal. The benefits of a third level of metal on Circuit Area and Delay are summarized in Figure. 20, where the comparison between the carry-strength adders routed on 2 and 3 levels of metal is shown in terms of area-delay product. As expected, for all the considered numbers of rows the layouts made using 3 levels of metal exhibit better characteristics than those made using only 2 levels.

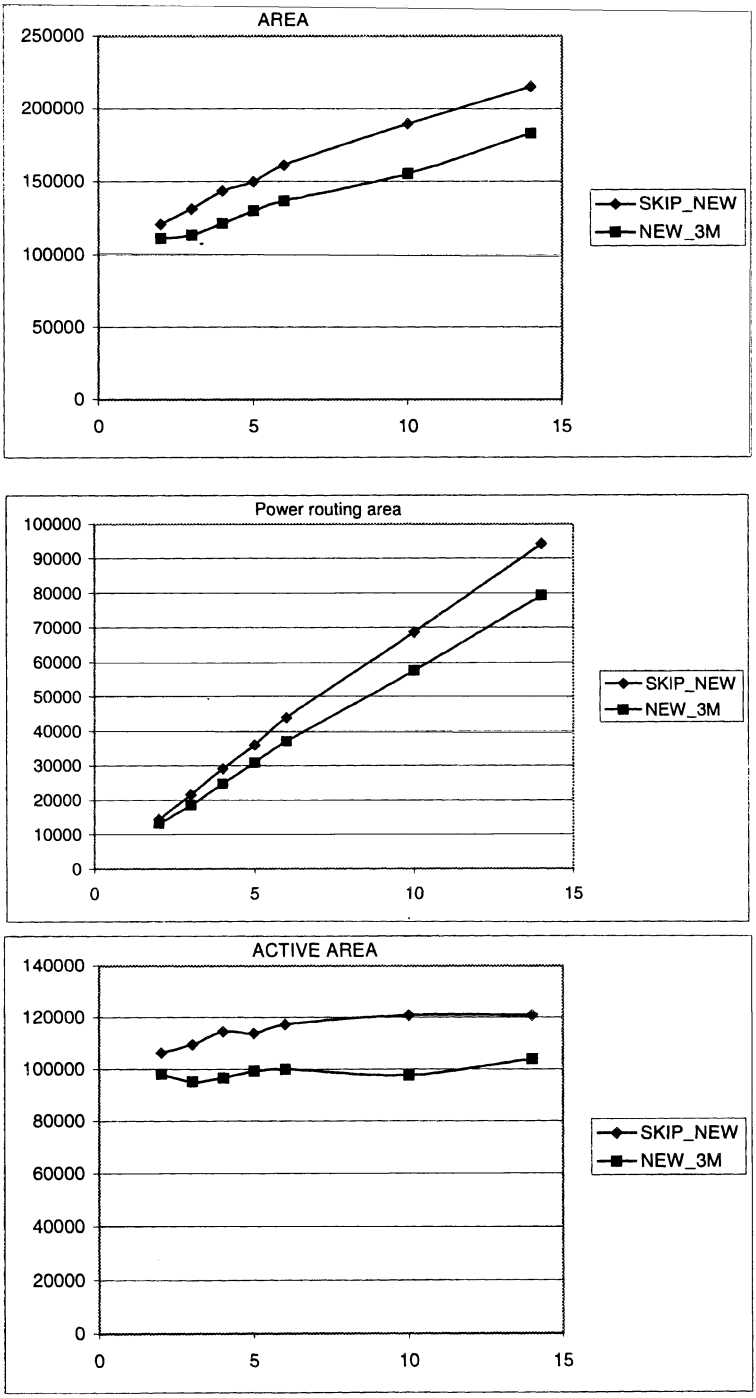


Figure 18: Area savings due to third level metal; *top:* (a) Total area, *bottom:* (b) Lateral power routing area, (c): Reduction of active area due to third level of metal

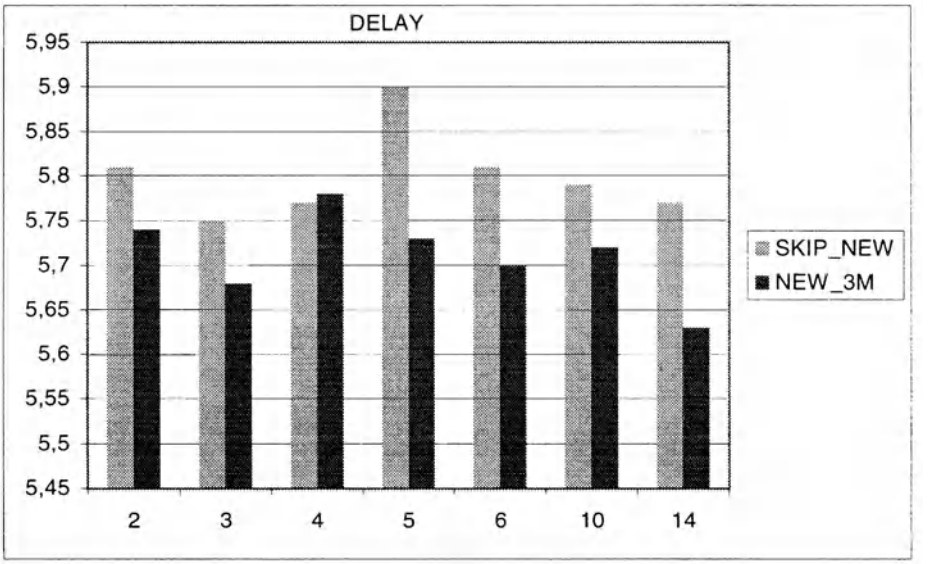


Figure 19: The reduction in the circuit delay varies as the number of rows changes and becomes more pronounced as the number of rows becomes very high

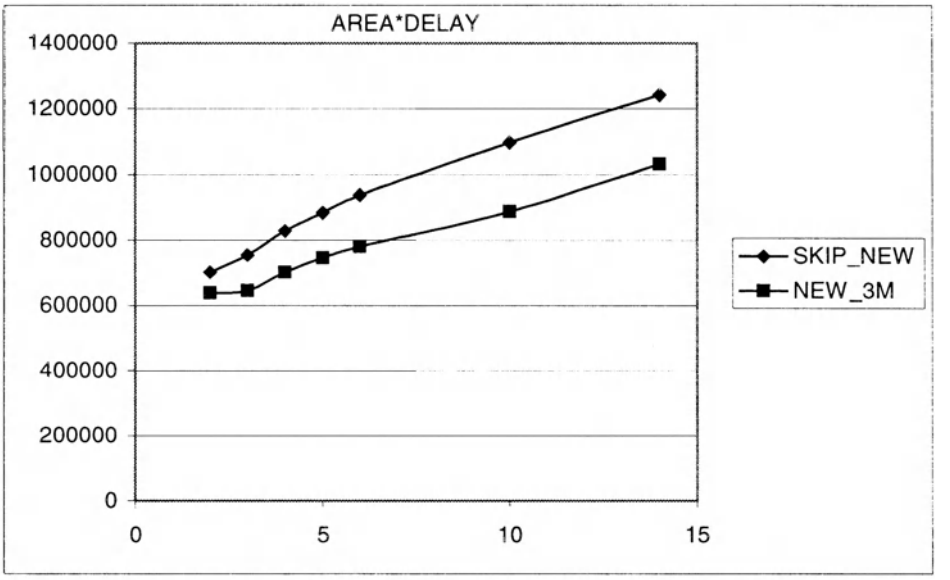


Figure 20: Comparison between the carry-strength adders routed using 2 and 3 levels of metal shown in terms of area-delay product

4 The Effects of Supply Voltage on Delay

Delay is one of the most important properties of a digital circuit since the majority of chip designs are limited more by speed than by area. Nevertheless, in the last few years many researchers have redirected their efforts to the realization of low-voltage circuits [5, 6, 7, 14] instead of the high-speed ones. This is due to the fact that reducing the supply voltage helps to alleviate power consumption problems. In fact, as it is well known, power consumption depends on the supply voltage by means of a quadratic relation. So that chosen a specific technology, reducing the supply voltage for example from 5V to 2.5V a reduction on the power consumption up to 70% can be expected.

Reducing the supply voltage does not significantly affects silicon area requirement, but it can compromise performance. This can be a serious problem in some applications such as digital signal processing in ASICs. Therefore we seek circuits that can operate at low voltage while still achieving adequate speed.

Normally the designer does not have too much control over the supply voltage, since it is determined by system and technology considerations. However, sometimes designers can choice a specific supply voltage in a given range. This is the case for the above-referred CUB and CUD processes, in which the supply voltage can range between 5V and 2V. This implies that also knowing how performance change versus the supply voltage has a fundamental importance to identify a reasonable area-delay-power trade-off.

In order to demonstrate how supply voltage changes can affect performance several electrical HSPICE (BSIM3V3 level 49) simulations have been carried out for all the adders described in section 2. To limit varying parameters in our experiments the number of rows of cells has been fixed to two for all the examined adders. A supply voltage changing from 5V to 2V with a step size equal to 1V has been imposed. In Figure. 21 the observed dependencies are summarized. Note that delay approximately triples for each adder as the supply voltage is dropped from 5V to 2V. In fact, the worst-case delay of the ripple-carry adder changes from 18ns at 5V to 50ns at 2V and the worst-case delay of the carry-skip adder changes from 10ns at 5V to 29ns at 2V! To better evaluate how the worst-case delay changes for the other adders the zoomed diagram reported in Figure. 22 can be examined. Since the carry-strength adder with optimized block sizes has the lowest delay at 5V to begin with (about 5ns), it can afford a reduction of the supply to 2V while keeping the delay down to about 15ns. Thus in terms of absolute delay increase, this adder is the best performer as the voltage decreases. This is

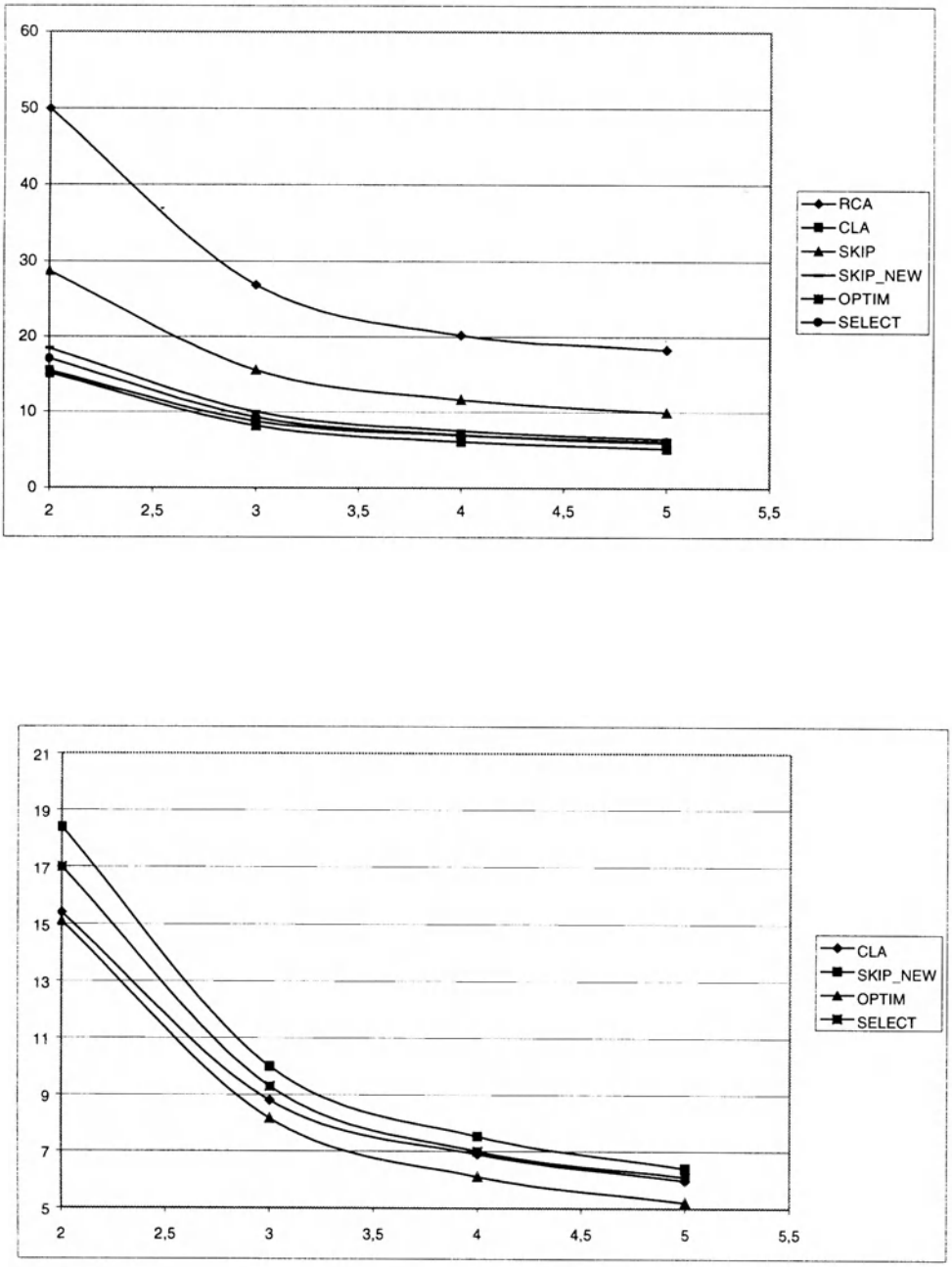


Figure 22: Dependency of delay on supply voltage, zoomed, without ripple carry adder

an important property not only because a better area-time-power trade-off can be achieved but also in terms of circuit robustness and reliability.

5 Conclusions

In this chapter we considered certain important tradeoffs in digital binary adder design. Such a study is important because adders lie at the core of a large number of digital systems. In particular, we consider the effects of floorplanning, number of levels of metals, and supply voltage on performance and area. One thing we found is that the layout area depends significantly on the aspect ratio, which means that traditional floorplanning algorithms can be improved to accommodate this dependence. As expected, circuit delay also depends on the aspect ratio. This dependence can be complex, and we have given it some explanation and supporting experimental data. We also showed how much the increase of the number of metal levels from two to three helps with respect to area and speed. Finally, we quantified how the supply voltage affects adder performance. In all these results we compared a new type of adders called "carry-strength" adder against traditional adders and found the new type of adders to be superior in all situations.

References

[1] A. P. Chandrakasan, S. Sheng, R. Brodersen "Low-power CMOS digital design," IEEE J. Solid-State Circuits, Vol.27, n 4, April 1992, pp.473-484.

[2] H. Esbensen and E. S. Kuh. "Explorer: An interactive floorplanner for design space exploration," EuroDAC-96, pp. 356-361, 1996.

[3] V. Kantabutra, "Designing Optimum Carry-Skip Adders," IEEE Trans. Computers, June, 1993.

[4] V. Kantabutra, P. Corsonello, and S. Perri, "Fast, low-cost adders using carry-strength signals," INVITED PAPER, SSGRR 2000, Computer and E-Business Conference, L'Aquila, Italy.

[5] U. Ko, P. T. Balsara, and W. Lee "Low-power design techniques for high-performance CMOS adders," IEEE Trans. VLSI Systems, Vol.3, n2, June 1995, pp.327-333

[6] M. Margala and N. G. Durdle, "Noncomplementary BiCMOS logic and CMOS logic for low-voltage, low-power operation - a comparative study," IEEE J. Solid State Circuits, Vol.33, n10, October 1998, pp.1580-1585

[7] C. Nagendra, R. M. Owens, and M. J. Irwin, "Power-delay characteristics of CMOS adders," IEEE Trans. VLSI systems, Vol.2, n3, September 1994, pp.377-381

[8] V. Narayananan, D. LaPotin, R. Gupta, and G. Vijayan, "Pepper – a timing driven early floorplanner," Proc. Int'l Conf. Computer Design, pp. 230 - 235, 1995.

[9] S. M. Sait, H. Youssef, S. Tanvir, and M. S. T. Benten. "Timing influenced general-cell genetic floorplanner," Proc. ASP-DAC '95; CHDL '95; VLSI '95; IFIP Int'l Conf. Hardware Description Languages; IFIP Int'l Conf. on VLSI, Asian and South Pacific, pp.135-140, 1995.

[10] N. A. Sherwani, Algorithms for VLSI Physical Design Automation, 3rd ed., Norwell, MA, Kluwer Academic Publisher, 1999, p. 193.

[11] W. Wolf Modern VLSI design - System on silicon, 2nd ed., Prentice Hall, Upper Saddle River NJ, USA, 1998

[12] H. Youssef, S. M. Sait, and K. J. Al-Farra, "Timing influenced force-directed floorplanning," Proc. EURO-DAC, pp. 156-161, 1995.

[13] T. Yamanouchi, K. Tamakashi, and T. Kambe. "Hybrid floorplanning based on partial clustering and module restructuring," IEEE/ACM Int'l Conf.,1995, pp. 478-483; ICCAD Digest of Tech. Papers, 1996;

[14] R. Zimmerman and W. Fichtner "Low-power logic style: CMOS versus pass-transistor logic," IEEE J. Solid-State Circuits, Vol.32, n7, July 1997, pp.1079-1090.