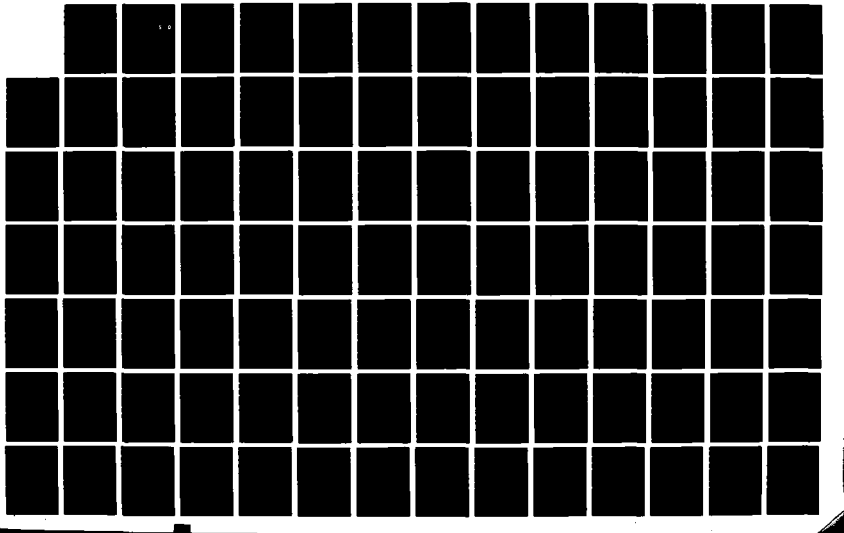
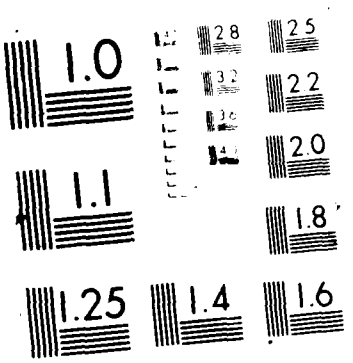
AD-A185 047 EXTRACTION OF MOS VLSI (VERY-LARGE-SCALE-INTEGRATED) 1/2
CIRCUIT MODELS INCLU..(U) ILLINOIS UNIV AT URBANA COLL
OF EDUCATION S SU SEP 87 UILU-ENG-87-2254
UNCLASSIFIED N00014-84-C-0149 F/G 9/1 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
2.5
2.2
2.0
1.8

September 1987 UILU-ENG-87-2254

COORDINATED SCIENCE LABORATORY
College of Engineering

DTIC FILE COPY

AD-A185 847

DTIC
ELECTE
S OCT 0 9 1987 D

EXTRACTION OF MOS VLSI CIRCUIT MODELS INCLUDING CRITICAL INTERCONNECT PARASITICS

Shun-Lin Su

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2254	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Ave. Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Joint Services Electronics Program 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-84-C-0149
8c. ADDRESS (City, State, and ZIP Code) 800 N. Quincy St. Arlington, VA 22217	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Extraction of MOS VLSI Circuit Models Including Critical Interconnect Parasitics

12. PERSONAL AUTHOR(S)
Su, Shun-Lin

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	September 1987	162

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
extraction, MOS VLSI, interconnection

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

As the feature sizes of Very-Large-Scale-Integrated (VLSI) circuits continue to decrease, the timing performance of a design cannot be estimated accurately without introducing the signal delay due to interconnect parasitics. Modeling interconnect parasitics directly from a circuit layout is therefore emphasized. In this research, two programs, FEMRC and HPEX, have been developed to investigate the following areas: (1) interconnect modeling, (2) hierarchical parasitic circuit extraction, and (3) collapsing technique for interconnects. The FEMRC is a two-dimensional, finite-element program which computes the resistance or the capacitance from the user-specified geometry. Since the equation formulation for FEMRC is based on a finite-element method, there is no shape restrictions on dielectric interfaces or conductor geometries. In resistance calculation, a quasi-three-dimensional effect of contact resistance is also taken into account. The program HPEX is a hierarchical parasitic circuit extractor which takes the CIF layout description as an input and generates a SPICE input with different details of interconnect parasitics. In this extractor, analytical formulas fitted from

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

19. Abstract (continued)

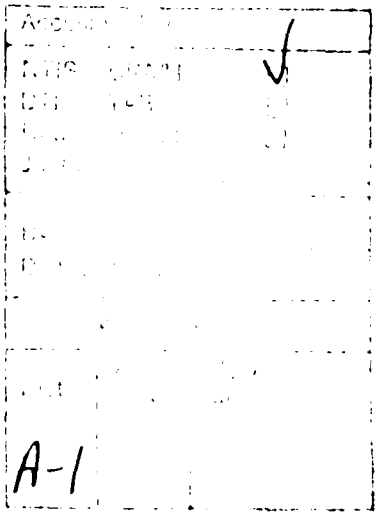
numerical data are used to model interconnect parasitics of VLSI circuits in order to compromise between the accuracy and the computation time. Simulations show that by carefully fitting data analytical formulas can be very accurate, especially when the interconnect region is fairly regular. Layout partition technique, which facilitates the interconnect modeling, is also studied and implemented in HPEX. Finally, a new node reduction technique which accurately reduces parasitic *RC* networks is studied. This technique is capable of reducing a large volume of interconnect data into a manageable size, thereby substantially reducing the effort needed in verification.

September 1987 UILU-ENG-87-2254

COORDINATED SCIENCE LABORATORY
College of Engineering

EXTRACTION OF MOS VLSI CIRCUIT MODELS INCLUDING CRITICAL INTERCONNECT PARASITICS

Shun-Lin Su



UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Approved for Public Release. Distribution Unlimited.

EXTRACTION OF MOS VLSI CIRCUIT MODELS INCLUDING CRITICAL INTERCONNECT PARASITICS

BY

SHUN-LIN SU

B.S.Eng., National Taiwan University, 1979
M.S., University of Illinois, 1983

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1987

Urbana, Illinois

EXTRACTION OF MOS VLSI CIRCUIT MODELS INCLUDING CRITICAL INTERCONNECT PARASITICS

Shun-Lin Su, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1987

As the feature sizes of Very-Large-Scale-Integrated (VLSI) circuits continue to decrease, the timing performance of a design cannot be estimated accurately without introducing the signal delay due to interconnect parasitics. Modeling interconnect parasitics directly from a circuit layout is therefore emphasized.

In this research, two programs, FEMRC and HPEX, have been developed to investigate the following areas: (1) interconnect modeling, (2) hierarchical parasitic circuit extraction, and (3) collapsing technique for interconnects. The FEMRC is a two-dimensional, finite-element program which computes the resistance or the capacitance from the user-specified geometry. Since the equation formulation for FEMRC is based on a finite-element method, there is no shape restrictions on dielectric interfaces or conductor geometries. In resistance calculation, a quasi-three-dimensional effect of contact resistance is also taken into account. The program HPEX is a hierarchical parasitic circuit extractor which takes the CIF layout description as an input and generates a SPICE input with different details of interconnect parasitics. In this extractor, analytical formulas fitted from numerical data are used to model interconnect parasitics of VLSI circuits in order to compromise between the accuracy and the computation time. Simulations show that by carefully fitting data analytical formulas can be very accurate, especially when the interconnect region is fairly regular. Layout partition technique, which facilitates the interconnect modeling, is also studied and implemented in HPEX. Finally, a new node reduction technique which accurately reduces parasitic *RC* networks is studied. This technique is capable of reducing a large volume of interconnect data into a manageable size, thereby substantially reducing the effort needed in verification.

ACKNOWLEDGEMENTS

I wish to express my most sincere gratitude to my advisor, Professor Timothy N. Trick, for his guidance, encouragement and patience during the course of this research. His continuing support and good advice are deeply appreciated.

I would also like to thank Professor Vasant B. Rao, my co-advisor, for many helpful discussions and suggestions in this research. Warmest thanks are extended to Professor Ibrahim N. Hajj and Professor Sung-Mo Kang for many fruitful discussions and for their encouragement. I am grateful to Professor P. Banerjee for being a member of my dissertation committee and for his suggestions. I am also indebted to all the members of the Circuits and Systems group for many enlightening conversations and for their friendship.

Finally, I would like to express my deepest appreciation to my family for their continual support and encouragement, and especially to my parents for their love and understanding. This thesis is dedicated to them.

TABLE OF CONTENTS

CHAPTER PAGE

CHAPTER 1.

INTRODUCTION

As the density and complexity of Very-Large-Scale-Integrated (VLSI) circuits continue to increase, detecting errors in circuit and layout designs manually becomes almost impossible. Hence, to ensure error-free circuit designs before being sent to the expensive fabrication step, it is important to have Computer-Aided-Design (CAD) tools to perform design verifications ranging from the behavior level to the circuit level. In general, extensive verification steps should be taken on the different levels of circuit designs to make sure that the functionality and performance of a VLSI system meet the user's specifications. A flowchart describing different levels of designs and their verifications is shown in Figure 1.1.

In order to handle the complexity problem inherent in VLSI designs, it is a common practice to design a VLSI system hierarchically. Once a VLSI system is constructed in this fashion, it is natural to exploit the design hierarchy in verification tools from the computer time point of view. Figure 1.1 also shows several levels of the hierarchy in the top-down design and their corresponding verifications. However, at each level of the top-down circuit design the functionality and performance of the system cannot be guaranteed unless feedback from physical designs is taken into consideration.

1.1. Functional and Performance Verification of a VLSI System

In verifying a VLSI system or circuit, two critical issues must be emphasized:

(1) functional verification and

(2) performance verification.

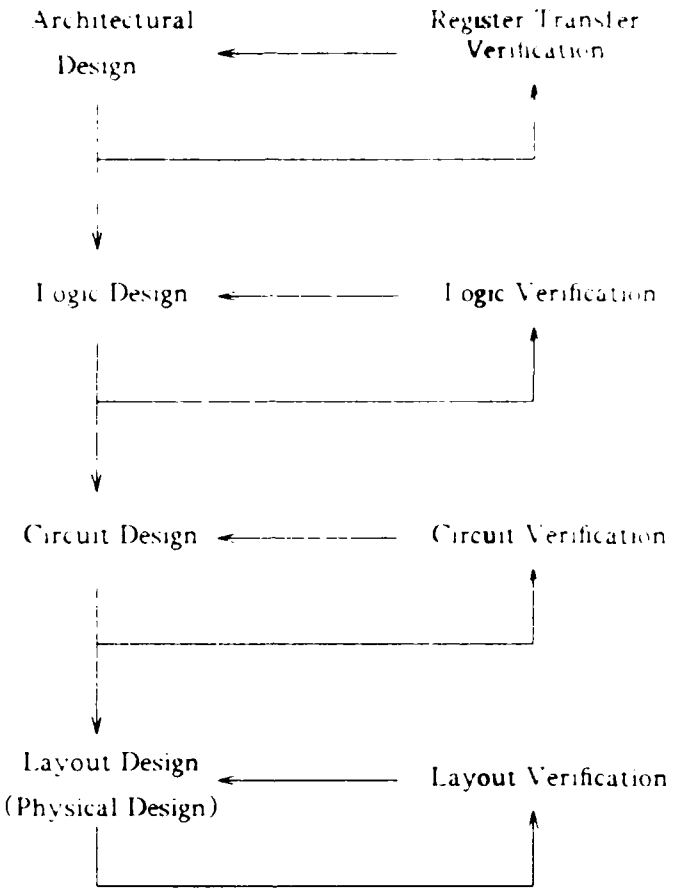


Figure 1.1 Flowchart for design verification.

Functional verification can be defined as verifying a design's functionality to ensure that the design performs the function as originally specified. Different levels of functional verification ranging from the behavior level to the circuit level are usually needed during the top-down design phase.

In each level of the functional verification, for the purpose of testing the functionality of a design, the designer should provide different input test vectors. Analysis programs, such as functional or logic simulators can be classified in this type of verification. However, if a design

[illegible] because [illegible]

[illegible] In other words, [illegible] the high level description [illegible] the designer need not verify the lower level functionality of the resulting design. This is one of the advantages of using a silicon compiler in the chip design.

The chief purpose of the performance verification for a VLSI chip is to ensure the required operating speed of this chip usually specified by the designer. The speed at which a design can operate is strongly dependent on the underlying process technology and the circuit design style. The circuit technology generally determines the inherent device delays, which in turn plays the most important part in the operating speed of a VLSI system. Therefore, the correct estimation of circuit element delays including both active and passive devices is a key factor for the successful performance verification of a chip design.

Since the lower level implementation has a significant impact on a VLSI system's speed, the correct abstraction from the lower level to a higher level delay model is imperative for the higher levels of performance verification. If a design synthesis tool such as a silicon compiler is utilized to directly generate the physical layout of the entire chip from the behavioral description, the silicon compiler has to perform timing estimation at various levels of the design in order to guarantee the speed requirement of the design. Because the delay concept is bottom-up and the silicon compilation is top-down, a timing-driven silicon compiler needs feedback from the lower levels to the higher level synthesis steps, unlike generating the correct functionality of a design.

1.2. Physical Parameter Extraction

One of the critical issues in the performance verification of the lowest level of a VLSI design is to correctly identify both transistors and interconnect parasitics in the physical layouts. This process is known as circuit extraction. Several programs [2-10] have been implemented to emphasize various aspects of circuit extraction depending on the verification tool to be used in the next phase, namely, the simulation phase. Basically, three types of information can be obtained by circuit extraction:

(1) circuit elements such as MOS devices, resistance and capacitances

(2) electrical parameters associated with circuit elements, and

(3) connectivity of circuit elements.

Types (1) and (3) are enough for circuit schematic comparison programs [11], and logic simulators [12], whereas all three types of information are required for circuit and switch-level timing simulators [13-14].

Previously, only active devices were extracted from the physical layout because they were sufficient to predict the circuit performance. But in the present-day device technology with submicron feature sizes, interconnect parasitics can no longer be neglected [15-17]. In order to be able to accurately predict delays through signal paths in VLSI circuits, resistances and capacitances associated with the interconnects should also be included in the output of circuit extraction. Since the actual value of interconnect capacitances and resistances is strongly dependent on the physical layout, it becomes important to find accurate models for interconnects in the circuit layout in order to correctly estimate the effect of interconnect parasitics on VLSI circuit performance.

The most difficult task in modeling interconnects is calculating interconnect resistances, self-capacitances and coupling capacitances between conductors. Recently, some techniques [3-

5, 10, 18-20] have been developed to model such interconnect parasitics from circuit layouts. One of them is to manipulate layout data first [18], then use a process simulator [21] to calculate parasitic electrical parameters. This technique can simulate the effect of process parameters on interconnect parasitics. However, it is very expensive, computationally, to simulate the fabrication process for VLSI circuits, even if analytical formulas are used in the process simulator. Another technique couples the library look-up method and the numerical method such as the finite difference method in modeling interconnects [3]. In this technique, the first step is to partition layout data into certain configurations. If the configuration is in the library, then it is not difficult to generate interconnect capacitance or resistance. On the other hand, if this configuration has not been encountered before, numerical methods need to be applied to calculate the interconnect parameters. This new configuration can then be added to the library. This technique gives quite accurate results because that the look-up tables are built by using accurate numerical methods. However, it is still time-expensive if the layout configuration is highly irregular

1.3. Features of HPEX

In this thesis, in order to demonstrate the detailed extraction of interconnect parasitics, a hierarchical, rectangle-based circuit extractor called HPEX (Hierarchical Parasitic and circuit EXtractor) has been developed. Features of HPEX can be described as follows.

1.3.1. Manhattan style layouts

Manhattan layouts described in the CIF [22] layout language are used as input to HPEX. In Manhattan layouts, all boundary segments are parallel to either x-axis or y-axis. There are two reasons to adopt this layout style in our study. First, the data structure and extraction algorithms can be simplified and made more efficient for Manhattan layouts during circuit extraction as compared to that for the nonrestricted layouts. Second, the Manhattan layout

style is gradually accepted in industry due to its simplicity in both manual and automatic layout designs. Therefore, in developing an experimental extractor such as HPEX, the Manhattan layout style is a good assumption.

1.3.2. Analytical formulas

Analytical formulas for resistance and capacitance computations along with a rectangle decomposition algorithm are employed in HPEX to model interconnect parasitics. The computation time can be reduced substantially, compared to using numerical methods to model interconnect parasitics. Although some accuracy might be lost with analytical formulas, it is still a good trade-off, considering the increasing complexity of layout designs.

1.3.3. Layout resizing

Layout resizing algorithms are applied to input layouts in the extraction preprocessing phase in order to compensate the process bias such as feature size widening due to lateral diffusion or shrinkage in polysilicon lines due to an incomplete photoresist developing. This consideration for extracting real conductor feature sizes will greatly enhance the accuracy of the parasitic modeling.

1.3.4. Separate process file

The process file for HPEX is separated from the execution code. A different process file can be specified without recompiling the program. Therefore, users can easily experiment with many different process files in order to optimize the design.

1.3.5. Different output files

Five different output files, as shown in Figure 1.2, can be generated by HPEX:

[1] Input file for schematic comparison program such as GEMINI [11]: If the designer wants to make sure that the layout has the same schematic as the original design, only the transistor list needs to be identified and fed into the schematic comparison program such as GEMINI, which takes the extracted and designed schematics as an input, then uses the graph isomorphism algorithm to test if these two schematics are the same. Some layout mistakes can be easily detected by this program.

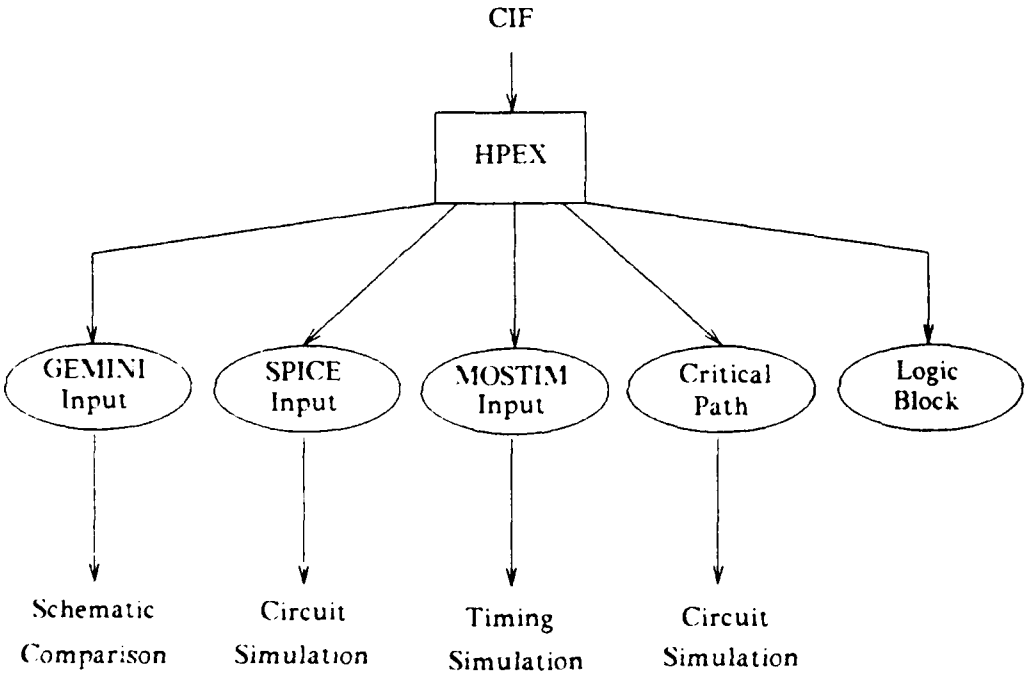


Figure 1.2 Different output files of HPEX.

[2] Input file for switch-level simulator such as MOSTIM [14]: If only rough timing information is required or a big circuit needs to be simulated, the extracted circuit should be suitable for the timing simulator or switch-level simulator such as MOSTIM. Since most of this type of simulators cannot handle interconnect resistances, the resistance extraction is not performed for this output file. In addition, all node capacitances including interconnect capacitances and device nonlinear capacitances are lumped to ground in the capacitance extraction because simple device models are generally employed in the timing or switch-level simulators.

[3] Input file for circuit simulator such as SPICE2 [13]: If the most accurate circuit simulation is required, the extracted output file should include all detailed interconnect parasitics such as self- and coupling capacitances, and all resistances. The nonlinear capacitances associated with MOS transistors are not extracted explicitly. Since SPICE can compute the device internodal capacitances for given device dimensions, only dimensions related to device capacitances are extracted. For example, in the transistor extraction the source area and perimeter are estimated.

[4] Input file containing only the critical path information for SPICE2 : A circuit simulator has its limitation in handling a circuit with a large number of transistors. One practical way to simulate a large chip is to simulate the critical path that determines the chip timing. By having user-specified nodes along the critical path, all transistors and interconnect parasitics in this path can be extracted by HPEX. Besides, the capacitive loading of all side branches is carefully estimated and added in the extracted file.

[5] Logic block description : In this output, the logic function of each gate is described in terms of boolean equation. At present, this type of extraction works only for conventional NMOS and CMOS circuit designs.

1.3.6. Node reduction

An accurate node reduction technique is developed and employed in HPEX to reduce the number of parasitic elements for the extracted SPICE input file. This will reduce the computation time in the SPICE simulation. In addition, the designer can easily pinpoint the interconnect parasitics which are responsible for the performance degradation if this degradation is indeed due to the interconnection delay.

1.4. Overview of Thesis

As mentioned in the previous section, interconnect modeling for VLSI circuits is a must in circuit performance verification when devices with small feature sizes are used. Modeling interconnect parasitics directly from a circuit layout, however, is getting difficult in terms of accuracy and computation time. In order to study the accuracy of parasitic modeling, two-dimensional empirical models for interconnect capacitance and resistance are first studied. Based on these models, a two-dimensional program called FEMRC is developed. Since the equation formulation for FEMRC is based on the finite-element method, there are no shape restrictions on dielectric interfaces for capacitance calculation. Not only capacitance but also resistance can be calculated by this program. In resistance calculation, a quasi-three-dimensional effect of contact resistance is also taken into account. Details on formulation and implementation of FEMRC are given in Chapter 2.

In Chapter 3, a layout partition technique for resistance and capacitance modeling similar to the one proposed by Mori and Wilmore in [20] is studied. By applying this technique, we use analytical formulas fitted from numerical data rather than numerical methods to model interconnect parasitics of VLSI circuits in order to compromise between the accuracy and the computation time. By carefully fitting data, analytical formulas can be very accurate, especially when the interconnect region is fairly regular. If a small part of the irregular layout

region requires accurate parasitic computation, we suggest using two- or even three-dimensional empirical models to calculate interconnect parasitics. Due to the complicated layout design, this extraction methodology compromising between efficiency and accuracy may be the best way to perform circuit extraction and to verify the design.

The data structure and geometrical algorithms used in HPEX are discussed in Chapter 4. In order to extract the actual feature sizes for interconnect modeling, an efficient layout resizing algorithm based on a scanline approach and a simple rectangle data structure is also presented in Chapter 4. Following this, extraction algorithms used in HPEX and details on their implementation are given in Chapter 5. Finally, a new node reduction technique which accurately reduces parasitic *RC* networks is studied. This technique is capable of reducing a large volume of interconnect data into a manageable size, thereby substantially reducing the effort needed in verification. The theoretical background and implementation details for the node reduction technique can be found in Chapter 6. Finally, conclusions and recommendations on future research topics are given in Chapter 7.

CHAPTER 2.

TWO-DIMENSIONAL MODELS FOR INTERCONNECTS

2.1. Introduction

Although it is well known that reduction of circuit dimensions will decrease the interconnect capacitance proportionately, there are two design considerations that force the capacitance to stay high. The first consideration is that designers try to increase the chip reliability with respect to the electromigration effect in aluminum conductors by scaling the conductor thickness down slowly in comparison to the reduction of feature size. The second consideration is that we attempt to minimize the signal delay by reducing the sheet resistance. Since the sheet resistance is inversely proportional to the conductor thickness, the latter thickness is also scaled down slowly compared to the chip horizontal scaling. As a result, the side-wall fringing capacitance of conductors becomes important in determining the overall capacitance. It has been predicted that the interconnection delay will become dominant in deciding the chip operating speed, especially when the scale-down of device feature size continues [15]. In order to accurately predict the performance of VLSI circuits, the capacitance of interconnect lines should be carefully modeled. Because of the important role played by the fringing field in determining the total capacitance, two-dimensional or even three-dimensional effects should be taken into account.

As mentioned in Chapter 1, not only the capacitance but also the resistance determines the performance of VLSI circuits. As for the resistance calculation, several methods have been reported [20, 23-26]. One of most commonly used methods computes the resistances of rectangular shaped conductors by simply counting the number of squares in the conductor and multiplying it by the sheet resistance. When using this method, the direction of current flow

must be estimated before calculating the length-to-width ratio of conductors. One disadvantage of this method is its limitation to the acceptable resistance shape. For example, the resistance shape with boundary segments not parallel with x-axis or y-axis cannot be correctly handled by this method. In addition, two-dimensional effects, such as current crowding, and contact resistance are not taken into account by this method. In order to improve resistance accuracy, numerical analyses of physical equations are generally applied. These numerical approaches include the Schwarz-Christoffel transformation [25], the finite difference method, and the finite-element method (FEM) [26]. In this chapter, the FEM is adopted to calculate the interconnect resistance. The proposed resistor model can handle any arbitrarily shaped, multi-region and multi-electrode conductor regions. Furthermore, by using a different physical equation for contact regions, contact resistances can be incorporated into the resistance calculation.

In the following sections, a two-dimensional model for calculating capacitances between multiple conductors is first described, followed by a discussion of two-dimensional resistance models. Then some features of a finite-element program called FEMRC are given. Finally, some examples of FEMRC for capacitance and resistance calculation are illustrated.

2.2. Capacitance of Multiconductors

2.2.1. Integral method

The integral method expresses the solution to the Laplace equation, which determines the potentials in a region **D**, by the following form:

$$\Phi(r) = \int_{\mathbf{D}} G(r,r')\sigma(r')dr' , \tag{2.1}$$

where $\Phi(r)$ is the potential, $\sigma(r')$ denotes the charge density at point r', and $G(r,r')$ is an appropriate Green function that describes the potential at r induced by a unit charge at r'. Although the indicated integration covers the entire region **D**, it needs in fact to be performed

only where the charge density $\sigma(r')$ is not zero. For an N-conductor system, if the potential at the surface of each conductor is specified, the only unknowns left in Eq. (2.1) are the charge densities. By using the method of moments and a set of basis and testing functions [27], charge densities in Eq. (2.1) can be solved by a system of linear equations.

Once charge densities are obtained on the surface of all conductors for different combinations of conductor potentials, the capacitance coefficients can be easily computed. Consider a three-conductor problem shown in Figure 2.1. The capacitance coefficients can be defined as

$$\begin{aligned} Q_1 &= C_{11}\Phi_1 + C_{12}(\Phi_1-\Phi_2) + C_{13}(\Phi_1-\Phi_3) \\ Q_2 &= C_{21}(\Phi_2-\Phi_1) + C_{22}\Phi_2 + C_{23}(\Phi_2-\Phi_3) \\ Q_3 &= C_{31}(\Phi_3-\Phi_1) + C_{32}(\Phi_3-\Phi_2) + C_{33}\Phi_3 \end{aligned} \tag{2.2}$$

where Q_i is the total charge on conductor i, Φ_j is the potential of conductor j, and C_{ij} denotes the capacitance coefficients. By applying even and odd mode potential patterns, C_{ij} in Eq. (2.2) can be easily determined.

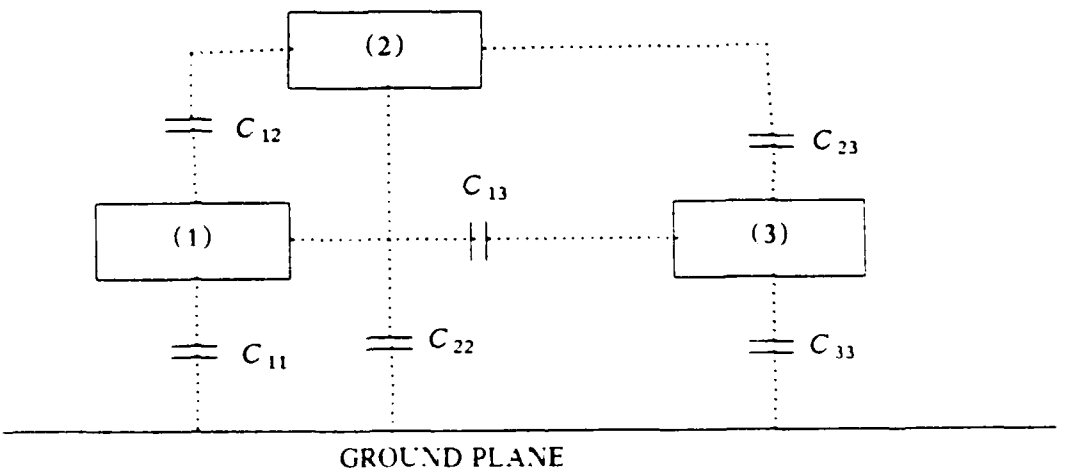


Figure 2.1 A three-conductor system.

One of the critical tasks in using integral method for capacitance calculation is to derive an appropriate Green's function for different dielectric regions. Two different methods for deriving Green's function have been proposed: (1) the method of images [28], and (2) spectral analysis [29]. However, these two methods are only practical for dielectric regions with planar interfaces. The Green function for non-planar dielectrics is too complicated to obtain explicitly. This is one of the disadvantages in applying the integral method to compute the capacitance. But from the computer time and storage point of view, the integral method is still a good and acceptable means to compute the interconnect capacitance. A typical program which uses this method to calculate capacitance is CAP2D [27].

2.2.2. Finite-element method

Instead of solving the charge densities, the differential method computes the potentials directly by solving the following Laplace equation over a region Ω subject to some boundary conditions.

$$\nabla\epsilon\nabla\Phi(x,y) = 0 \qquad (x,y) \in \Omega \tag{2.3}$$

Boundary condition (BC) has to be specified at every point of the boundary in order to obtain a unique solution of the above equation. Boundary conditions can be further classified into two principal types for the Laplace equation:

(1) *Dirichlet (or Essential) BC*: An equation relating the values of Φ at points of the boundary

(2) *Neumann (or Natural) BC*: An equation relating the values of first derivative of Φ ($\nabla\Phi$) at points of the boundary.

Two methods can be employed to numerically solve the Laplace equation: (1) finite-difference method (FDM), and (2) finite-element method (FEM). The approach we describe here is the

finite-element method, which is more flexible in handling irregular boundaries and nonplanar dielectrics.

Two methods are generally used to formulate finite-element equations: (1) minimization of the functional, and (2) weighted residual process. Since the functional of the Laplace equation is explicit, it is convenient to apply the minimization of the functional to derive finite-element equations. The problem to solve Eq. (2.3) is then equivalent to finding Φ, which satisfies boundary conditions and minimizes

$$\chi = \int\int_{\Omega} \frac{1}{2}\epsilon \left[\left| \frac{\partial \Phi}{\partial x} \right|^2 + \left| \frac{\partial \Phi}{\partial y} \right|^2 \right] dx dy \ , \tag{2.4}$$

where Ω is the region of interest. If the region of interest is discretized into N grid points, the approximate potential in this region can be expressed as

$$\Phi(x,y) = \sum_{i=1}^{N} N_i(x,y)\phi_i \ , \tag{2.5}$$

where N_i is the shape function (or the interpolation function), and ϕ_i is the potential at node i. The shape function N_i has the following property:

$$N_i(x_j,y_j) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{2.6}$$

For a linear triangular element shown in Figure 2.2, the shape function can be derived as

$$N_i(x,y) = \frac{(a_i + b_i x + c_i y)}{2\Delta} \tag{2.7}$$

where the area Δ is given by

$$\Delta = \frac{1}{2} \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}, \tag{2.8}$$

and the constants a_i, b_i, and c_i are given in terms of nodal coordinates by

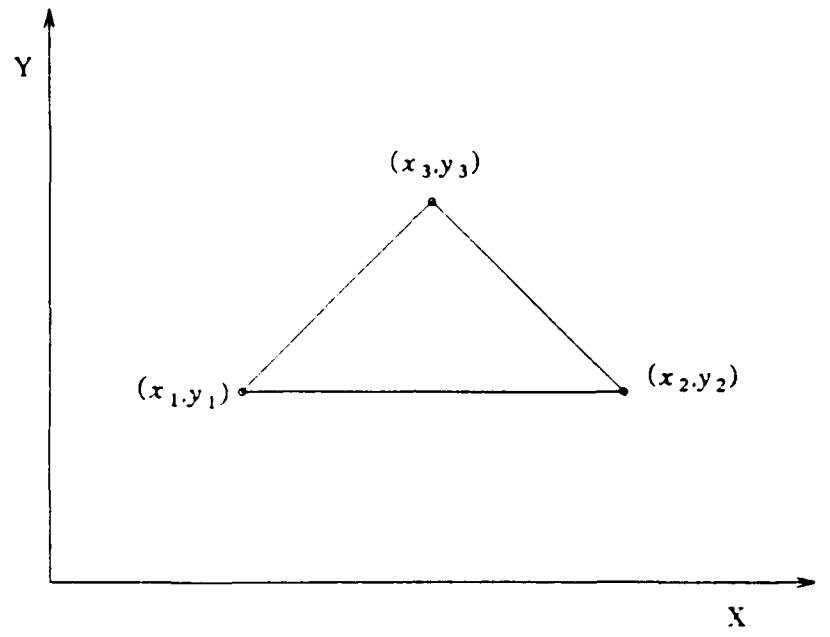


Figure 2.2 A triangular element.

$$a_1 = x_2y_3 - x_3y_2$$

$$b_1 = y_2 - y_3$$

$$c_1 = x_3 - x_2 , \tag{2.9}$$

with the others being obtained by cyclic permutation. Let the unknown potential Φ be defined element by element; the element contribution could then be evaluated using Eq. (2.4). For any node i, $i = 1,2,3$, by differentiating Eq. (2.4), we can write

$$\frac{\partial \chi^e}{\partial \phi_i} = \int\int_{\Omega(e)} \epsilon \left| \frac{\partial \Phi}{\partial x} \frac{\partial}{\partial \phi_i} \left| \frac{\partial \Phi}{\partial x} \right| + \frac{\partial \Phi}{\partial y} \frac{\partial}{\partial \phi_i} \left| \frac{\partial \Phi}{\partial y} \right| \right| dx dy \tag{2.10}$$

Substituting Eq. (2.5) into Eq. (2.10), We have the following matrix equation for a triangular element:

$$\frac{\partial \chi^e}{\partial [\phi]^e} = [h]^e [\phi]^e , \tag{2.11}$$

where the element (or stiff) matrix $[h]^e$ for a linear triangular element is given by

$$[h]^e = \frac{\epsilon}{4\Delta}\begin{vmatrix} b_1b_1+c_1c_1 & b_1b_2+c_1c_2 & b_1b_3+c_1c_3 \\ b_2b_1+c_2c_1 & b_2b_2+c_2c_2 & b_2b_3+c_2c_3 \\ b_3b_1+c_3c_1 & b_3b_2+c_3c_2 & b_3b_3+c_3c_3 \end{vmatrix} \tag{2.12}$$

It may be noted that the element matrix is symmetric.

After assembling a whole set of minimization equations and imposing suitable boundary conditions such as conductor potentials, we can solve the following set of linear equations to find the potential at each grid point:

$$[H]\{\phi\} = \{b\} \tag{2.13}$$

In the above equation, the right-hand vector $\{b\}$ is nonzero because of Dirichlet boundary conditions being forced. In this case, the Dirichlet boundary condition is the conductor voltage. In order to get a better accuracy in FEM, a typical global matrix $[H]$ which contains a large number of equations is used. However, in general, most of the terms in the matrix $[H]$ are zero. Therefore, sparse matrix techniques such as minimum fill-in reordering and LU factorization are employed to solve Eq. (2.13) in order to save memory space and CPU time.

Once potentials at grid points are obtained by Eq. (2.13), the electric field within each linear triangular element can be computed as follows:

$$\vec{E} = -\nabla\Phi = \frac{1}{2\Delta}\left\{(b_1\phi_1+b_2\phi_2+b_3\phi_3)\bar{x} + (c_1\phi_1+c_2\phi_2+c_3\phi_3)\bar{y}\right\}, \tag{2.14}$$

where $\bar{x}$ and $\bar{y}$ are unit vectors in the x- and y-directions, respectively. By integrating the electric field around the conductors and applying the Gauss law, the charge on each conductor is given by

$$Q = \int_c \epsilon\vec{E}\, d\bar{\tau} \tag{2.15}$$

Considering an N-conductor system, the capacitance matrix can be solved by the following matrix equation:

$$[C] = [Q][V]^{-1}. \tag{2.16}$$

where $[V]$ is the voltage excitation matrix which contains N independent excitations, and $[Q]$ is the corresponding charge matrix.

Since it is impossible to discretize the region with open boundary in the FEM, a closed region is used to approximate the infinite region. As a consequence, some accuracy may be lost due to this approximation. However, by carefully choosing the region of importance, we still can maintain the accuracy of results. For example, in capacitance calculation the closed boundary is chosen such that the electric field on the boundary should be negligible.

As described before, unlike the integral method, the finite-element method is not restricted to the geometric configuration of the insulators and conductors. This has an important implication in the present multilayered interconnect technology because the true structure of conductors and surrounding insulating layers can be simulated. Therefore, in a VLSI chip with small feature sizes, the finite-element method seems to be a better approach in determining the interconnect capacitance.

2.3. Resistance Calculation

2.3.1. Conductor region without contact windows

Consider an interconnect region Ω with no contact windows as shown in Figure 2.3. Let the electrodes be around the boundary of this region. The governing equation can be described by the following Laplace equation:

$$\nabla \frac{1}{R_s} \nabla \Phi(x,y) = 0 \quad (x,y) \in \Omega \tag{2.17}$$

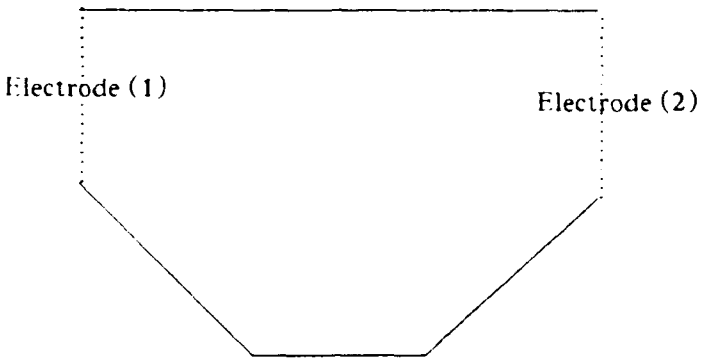


Figure 2.3 A conductor region without contacts.

Boundary Conditions:

$$\Phi(x,y) = Constant \quad (x,y) \in Electrode\ (1) \text{ and } Electrode\ (2)$$

$$\nabla\Phi(x,y) = 0 \quad (x,y) \notin Electrode\ (1) \text{ and } Electrode\ (2)$$

where R_s is the sheet resistance. Since Eq. (2.17) has the same form as Eq. (2.3), two-dimensional resistance calculation in this case follows the same procedures described in the preceding section. The only difference is that the permittivity ϵ is replaced by $\frac{1}{R_s}$. The element matrix for a linear triangular element in resistance calculation is given by

$$[h]^e = \frac{1}{4R_s\Delta}\begin{vmatrix} b_1b_1+c_1c_1 & b_1b_2+c_1c_2 & b_1b_3+c_1c_3 \\ b_2b_1+c_2c_1 & b_2b_2+c_2c_2 & b_2b_3+c_2c_3 \\ b_3b_1+c_3c_1 & b_3b_2+c_3c_2 & b_3b_3+c_3c_3 \end{vmatrix} \tag{2.18}$$

After the global matrix is assembled from all element matrices by incorporating the boundary conditions, the potentials at grid points can be solved. From the potential information, the current flowing out or into each electrode can be easily calculated by

$$I_e = \int_l \vec{J}\, d\vec{s} = -\frac{1}{R_s}\int_l \nabla\Phi \cdot d\vec{s} \tag{2.19}$$

where Γ is a line segment along the electrode E. Because a linear interpolation potential function has been assumed inside a linear triangular element, the current density in a triangular region is a constant.

Consider an interconnect region with N electrodes. The resistances between N electrodes can be calculated by repeatedly applying the following procedures:

(1) For electrode i, set $\Phi_i = 1$ and $\Phi_j = 0\ \forall\, j \neq i$.

(2) Obtain potential solution and calculate the electrode current I_j.

(3) Resistance R_{ij} is then computed by the following equation:

$$R_{ij} = \frac{\Phi_i}{I_j} = \frac{1}{I_j}. \tag{2.20}$$

Repeat the above procedures for i from 1 to N, a complete resistance network with $\frac{N(N+1)}{2}$ resistors can be obtained.

2.3.2. Conductor region with contact windows

Due to a large amount of contacts required in a VLSI chip design, the metal-to-diffusion and metal-to-polysilicon contact resistances inevitably play an important role in determining the total interconnect resistance. In this section, we will take the contact windows into the finite-element equation formulation in order to calculate the interconnect resistance including the contact resistance. Consider a diffusion region with boundary and contact window electrodes as shown in Figure 2.4. Assume that the contact regions are covered by metal and the metal voltage is constant. For this case, the two-dimensional field problem can be described [30] as follows:

(1) Outside the contact window, the potential must satisfy the Laplace equation, namely, Equation (2.17).

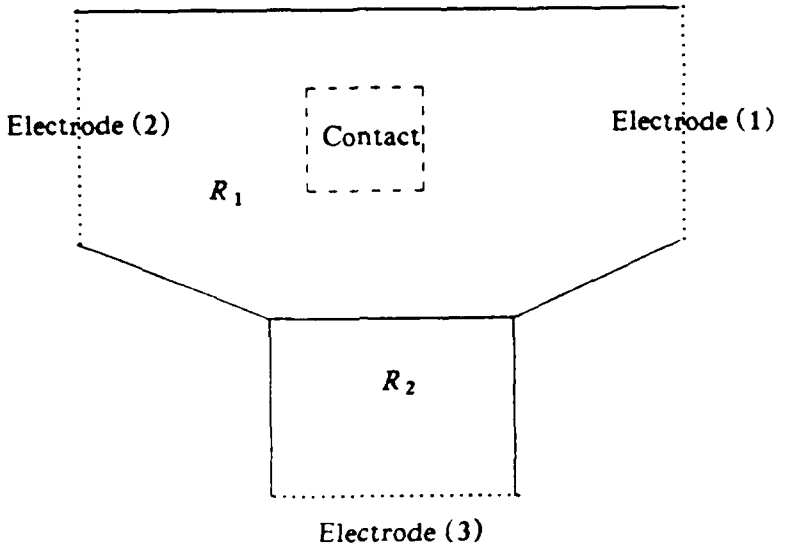


Figure 2.4 A conductor region with an inside contact.

(2) Inside the contact window, the potential is governed by

$$\nabla\left(\frac{1}{R_s}\nabla\Phi\right) = \frac{1}{\rho_c}\left(\Phi-\Phi_m\right), \tag{2.21}$$

which is the Helmholtz equation, where ρ_c is the specific contact resistivity and Φ_m is the voltage on the contact metal. The specific contact resistivity ρ_c is defined as

$$\rho_c = \left(\frac{\partial J(v_{ms})}{\partial v_{ms}}\right)^{-1}_{v_{ms}=0}. \tag{2.22}$$

where v_{ms} is the potential difference between the metal Fermi potential and the semiconductor potential. For each process, ρ_c is unique and can be determined by a certain test structure.

The case of calculating resistances for the regions outside the contact window has already been discussed in the previous section. In this section we will concentrate on formulating the finite-element equations in the region within the contact window. It has been shown that the

functional of the Helmholtz equation is

$$\chi = \iint_{\Omega} \left| \frac{1}{2R_s} \left(\left| \frac{\partial \Phi}{\partial x} \right|^2 + \left| \frac{\partial \Phi}{\partial y} \right|^2 \right) + \frac{1}{2\rho_c} \Phi^2 - \frac{1}{\rho_c} \Phi_m \Phi \right| dx dy \ . \tag{2.23}$$

If again the linear triangular element is chosen, by differentiating the elemental functional with respect to nodal potentials, we have the following matrix equation:

$$\frac{\partial \chi^e}{\partial \{\phi\}^e} = [h]^e \{\phi\}^e + \{F\}^e \ . \tag{2.24}$$

where

$$[h]^e = \frac{1}{4R_s \Delta} \begin{bmatrix} b_1b_1+c_1c_1 & b_1b_2+c_1c_2 & b_1b_3+c_1c_3 \\ b_2b_1+c_2c_1 & b_2b_2+c_2c_2 & b_2b_3+c_2c_3 \\ b_3b_1+c_3c_1 & b_3b_2+c_3c_2 & b_3b_3+c_3c_3 \end{bmatrix} + \frac{\Delta}{12\rho_c} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}. \tag{2.25}$$

and

$$\{F\}^e = -\frac{\Phi_m \Delta}{3\rho_c} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{2.26}$$

Consider a conductor with both noncontact and contact regions. The global matrix equation associated with this conductor can be assembled on an element-by-element basis. In a matrix assembling process, two different types of element matrices should be applied depending on the location of the element. Equation (2.18) is used for a noncontact element, while both Eqs. (2.25) and (2.26) are applied for a contact element. Using the same procedure as described in the previous section, a complete resistance network corresponding to the region of interest can be obtained. It is noted that the current flowing into a contact window can be calculated either by integrating the current density around the contact window or by summing the current contributed by all the elements inside the contact region. Using the latter method, we can derive the current flowing into a contact in a closed form:

$$I = \sum_{i \in Contact} I_i$$

$$= \sum_{i \in Contact} \iint_{e_i} \frac{1}{\rho_c} \left\{ \Phi_m - \left[N_{i1}\phi_{i1} + N_{i2}\phi_{i2} + N_{i3}\phi_{i3} \right] \right\} dxdy$$

$$= \sum_{i \in Contact} \frac{1}{\rho_c \Delta} \left\{ \Phi_m - \frac{1}{3} \left[\phi_{i1} + \phi_{i2} + \phi_{i3} \right] \right\} . \tag{2.27}$$

where e_i is the region of the triangular element i.

2.4. Finite-Element Program : FEMRC

A finite-element program FEMRC, which implements the previously described capacitance and resistance models, has been developed. The main flowchart showing the capacitance and resistance calculation phases is illustrated in Figure 2.5. The grid generator IGGI2 from the device simulator PISCES-IIB [31] is employed in FEMRC to automate the preliminary triangulation phase by the user-specified boundary points and regions.

The main features of FEMRC are

(1) Due to the use of IGGI2 the users only have to provide a small amount of input data compared to the massive amount of data for all triangular elements required in computation.

(2) Having different options, FEMRC can simulate both two-dimensional capacitances and resistances.

(3) In capacitance calculation, nonplanar dielectrics are easily handled. There is no limitation on the number of conductors or dielectrics.

(4) In the resistance calculation, any arbitrarily shaped, multiregion and multielectrode resistor can be simulated.

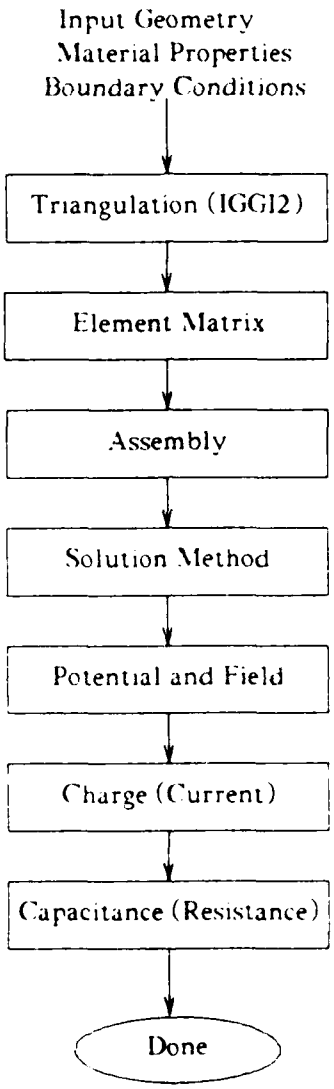


Figure 2.5 Flowchart for FEMRC.

The above features indicate that FEMRC is a useful tool in estimating VLSI interconnect capacitances and resistances.

2.5. Examples and Discussion

Example 1:

Figure 2.6 shows the cross section of an interconnect structure used for capacitance calculation, wherein three metal conductors covered by two levels of dielectrics are situated on the top of silicon dioxide. The thickness of the metal is $1\mu m$ and the thickness of field oxide is $1.83\mu m$. Various combinations of dielectrics, conductor widths and spacings listed in Table 2.1 have been employed for simulation. In the first set of the simulations, planar dielectric interfaces are assumed in order to compare the simulation accuracy for FEMRC and CAP2D. The simulation results are listed in Tables 2.2 and 2.3 for self-capacitance and coupling capacitance, respectively. The self-capacitance in this case is the capacitance between the central line and the substrate (or the ground plane), while the coupling capacitance is the capacitance between the central line and either of the adjacent lines. The capacitance C_m is the measured data taken

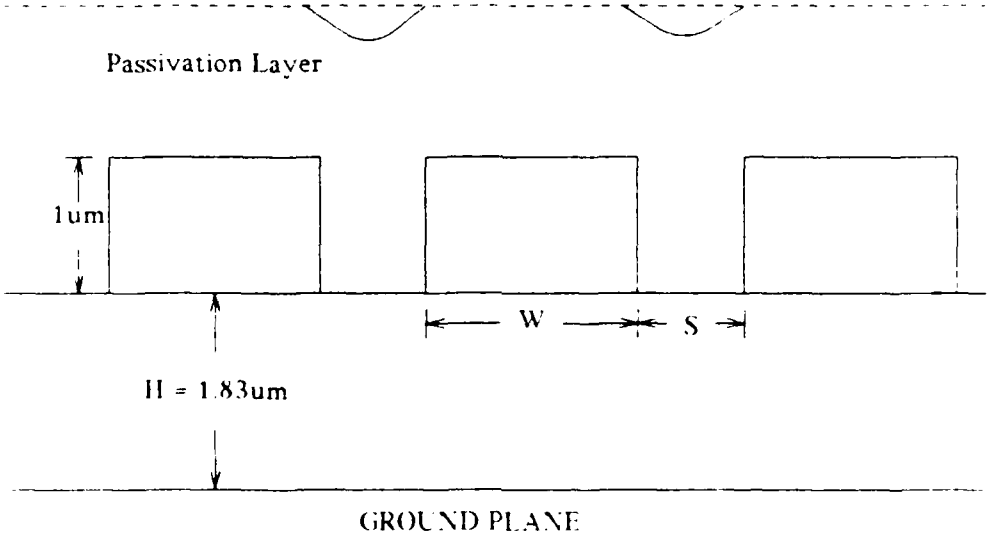


Figure 2.6 Cross section of three interconnection lines.

Table 2.1 Test structures and different passivations for FEMRC calculations.

test structures			
case	pitch (μm)	linewidth W (μm)	spacing S (μm)
(1)	3	1.7	1.3
(2)	4	1.8	2.2
(3)	5	2.3	2.7
(4)	5	2.8	2.2
passivations			
(A)	No passivation ($\kappa_{vacuum} = 1$).		
(B)	1 μm thick SiO_2 ($\kappa_{SiO_2} = 3.9$).		
(C)	1 μm thick nitride ($\kappa_{nitride} = 7.0$).		
(D)	1 μm thick SiO_2 + 1 μm thick nitride on the top.		

[†]C. P. Yuan and T. N. Trick, ICCAD-84, pp. 263-265.

Table 2.2 Percentage errors of self-capacitances calculated by FEMRC and CAP2D for the structures listed in Table 2.1[†].

passivation		case (1)	case (2)	case (3)	case (4)
(A)	C_m	0.05	0.063	0.077	0.081
	$C_s(FEMRC)$	0.0541	0.0625	0.0796	0.0816
	$\%(FEMRC)$	8.2	-0.79	3.4	0.74
	$\%(CAP2D)$	10	3.2	1.3	3.7
(B)	C_m	0.055	0.071	0.088	0.091
	$C_s(FEMRC)$	0.056	0.0673	0.0893	0.0884
	$\%(FEMRC)$	1.8	-5.2	1.5	-2.9
	$\%(CAP2D)$	4.5	0.6	-1.8	-0.9
(C)	C_m	0.058	0.076	0.094	0.096
	$C_s(FEMRC)$	0.0568	0.0695	0.0922	0.0904
	$\%(FEMRC)$	-2.1	-8.6	-1.9	5.8
	$\%(CAP2D)$	0.7	-2.6	-4.3	-4.2
(D)	C_m	0.057	0.074	0.091	0.094
	$C_s(FEMRC)$	0.0578	0.0687	0.0905	0.0896
	$\%(FEMRC)$	1.4	-7.2	-0.55	-4.9
	$\%(CAP2D)$	6.5	-0.3	-2.1	-2.3

[†]Assume planar dielectric interfaces for both FEMRC and CAP2D calculations.

[•]Percentage errors calculated by CAP2D are taken from C. P. Yuan and T. N. Trick, ICCAD-84, pp. 263-265.

Table 2.3 Percentage errors of coupling capacitances calculated by FEMRC and CAP2D for the structures listed in Table 2.1[†].

passivation		case (1)	case (2)	case (3)	case (4)
(A)	C_m	0.0205	0.0130	0.0107	0.0145
	C_c (*FEMRC*)	0.0172	0.0097	0.0075	0.0104
	% (*FEMRC*)	-16	-25	-30	-28
	% (*CAP2D*)*	-8.8	-19	-21	-23
(B)	C_m	0.049	0.030	0.024	0.031
	C_c (*FEMRC*)	0.0485	0.0291	0.023	0.0294
	% (*FEMRC*)	-1.0	-3.0	-4.2	-5.2
	% (*CAP2D*)	3.9	2.0	2.9	1.9
(C)	C_m	0.070	0.039	0.030	0.040
	C_c (*FEMRC*)	0.0814	0.050	0.0399	0.0498
	% (*FEMRC*)	17	28	33	25
	% (*CAP2D*)	21	33	40	33
(D)	C_m	0.050	0.0345	0.0268	0.0362
	C_c (*FEMRC*)	0.0529	0.0341	0.0287	0.0357
	% (*FEMRC*)	5.8	1.2	7.1	-1.4
	% (*CAP2D*)	9.6	2.6	13	4.4

[†]Assume planar dielectric interfaces for both FEMRC and CAP2D calculations.

*Percentage errors calculated by CAP2D are taken from C. P. Yuan and T. N. Trick, ICCAD-84, pp. 263-265.

from [32]. Percentage errors for both FEMRC and CAP2D by using the measured data as the reference are computed as follows:

$$Percentage\ error\ (\%) = 100\times\frac{C_c - C_m}{C_m}.$$

By comparing the results, it is observed that percentage errors for self- and coupling capacitance calculations are comparable for FEMRC and CAP2D if the measured data are used as the reference. The percentage errors for self-capacitance are all within 10%. However, the percentage errors for coupling capacitance are consistently greater, especially in passivations (A) and (C). This may be due to the fact that coupling capacitance is highly sensitive to the variation of

conductor spacing and dielectric between conductors. Besides, it should be noted that the measured data on a physical model of a structure cannot be regarded as exact

In order to take a step coverage of passivation into account, a piecewise line is used in the second set of the simulations to approximate the nonplanar dielectric interface in the passivation layer. The simulation results along with the measured data are listed in Tables 2.4 and 2.5. It should be noted that passivation (A) is not included in the tables because no dielectric covers the conductors in this case. By comparison, we can see that the percentage error for self-capacitance stays almost the same. However, the results for the coupling capacitance have been improved on the average, for example, in the case of passivations (C) and (D). From this observation, it is indicated that if we want to improve the simulation accuracy by taking nonplanar dielectric into account, FEMRC is an indispensable tool.

Table 2.4 Percentage errors of self-capacitances calculated by FEMRC for the structures listed in Table 2.1[†].

passivation		case (1)	case (2)	case (3)	case (4)
(B)	C_m	0.055	0.071	0.088	0.091
	C_s (*FEMRC*)	0.056	0.0679	0.0827	0.0874
	% (*FEMRC*)	1.8	-4.4	-6.0	-4.0
(C)	C_m	0.058	0.076	0.094	0.096
	C_s (*FEMRC*)	0.0567	0.0702	0.0860	0.0897
	% (*FEMRC*)	-2.2	-7.6	-8.5	-6.6
(D)	C_m	0.057	0.074	0.091	0.094
	C_s (*FEMRC*)	0.0577	0.0694	0.0842	0.0888
	% (*FEMRC*)	1.2	-6.2	-7.5	-5.5

[†]Assume nonplanar dielectric interfaces for FEMRC calculations.

Table 2.5 Percentage errors of coupling capacitances calculated by FEMRC for the structures listed in Table 2.1[†].

passivation		case (1)	case (2)	case (3)	case (4)
(B)	C_m	0.049	0.030	0.024	0.031
	C_c (*FEMRC*)	0.0471	0.0251	0.0195	0.0263
	% (*FEMRC*)	-3.9	-16	-19	-15
(C)	C_m	0.070	0.039	0.030	0.040
	C_c (*FEMRC*)	0.0778	0.0401	0.0309	0.0414
	% (*FEMRC*)	11	2.8	3.0	3.5
(D)	C_m	0.050	0.0345	0.0268	0.0362
	C_c (*FEMRC*)	0.0521	0.0333	0.0277	0.0354
	% (*FEMRC*)	4.2	-4.3	3.4	-2.2

[†]Assume nonplanar dielectric interfaces for FEMRC calculations.

Example 2:

In this example, two conductors with slant sidewalls situated on the top of oxide, as shown in Figure 2.7, are simulated. From a cross-sectional view of this structure, it can be seen that the conductors are first covered by a layer of nitride with $0.35\mu m$ thickness, then another thick layer of polyimide. Simulation results for conductors with spacing ranging from $1.0\mu m$ to $5.5\mu m$ are listed in Table 2.6. The capacitance C_f is the fringing capacitance between the conductor edge and ground, and C_c is the coupling capacitance between two conductors. In Table 2.6, calculated capacitances for both nonplanar and planar dielectric cases are both listed for comparison. It is observed that the nonplanar dielectric interface in general has greater impact on the coupling capacitance than on the fringing capacitance. This is because the non-planar dielectric interface significantly changes the equivalent dielectric constant between two conductors if there is a large difference between the dielectric constants of the two layers of passivations. Without considering the nonplanar dielectric effect, we obtain an average 15%

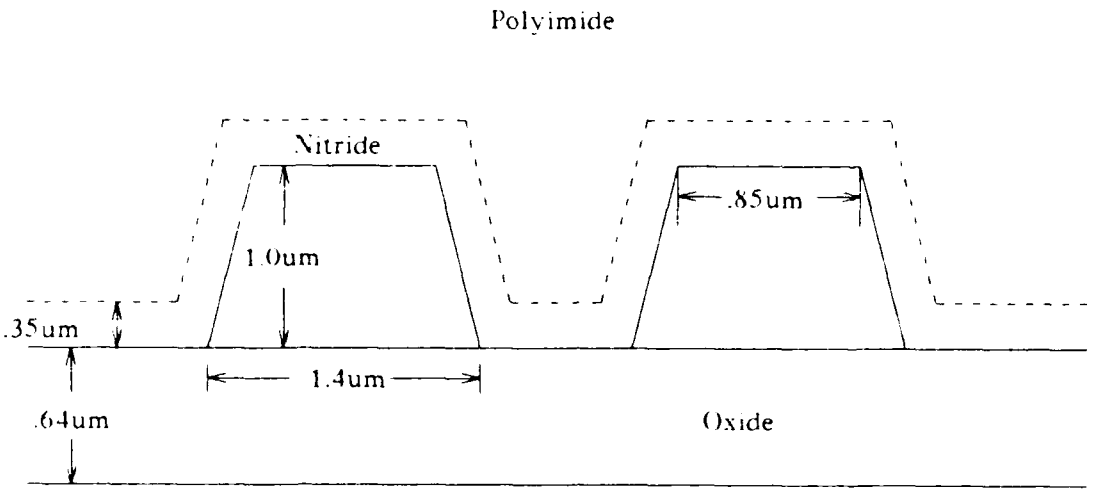


Figure 2.7 Cross-sectional view of two conductors with a nonplanar dielectric interface.

Table 2.6 Comparison of capacitances calculated by FEMRC for the structure shown in Figure 2.7 with different dielectric interfaces.

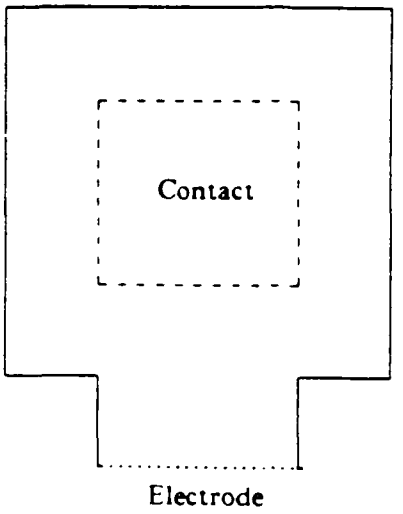
Separation (μm)	Nonplanar dielectric		Planar dielectric	
	C_s	C_c	C_s	C_c
1.0	0.0217	0.0579	0.0219	0.0706
1.5	0.0288	0.0377	0.0302	0.0495
2.0	0.0346	0.0270	0.0375	0.0343
2.5	0.0405	0.0201	0.0432	0.0275
3.0	0.0437	0.0155	0.0507	0.0199
3.5	0.0465	0.0121	0.0513	0.0170
4.0	0.0492	0.0096	0.0558	0.0124
4.5	0.0508	0.0077	0.0593	0.0098
5.0	0.0522	0.0062	0.0594	0.0079
5.5	0.0536	0.0050	0.0637	0.0064

[†]P. E. Cottrell and E. M. Buturla, IBM J. Res. Develop., pp. 277-288, May 1985.

error in the fringing capacitance calculation. However, the percentage error is increased to 25% for the case of the coupling capacitance calculation.

Example 3:

Figure 2.8 shows a diffusion region with one contact electrode and one boundary electrode. Assume that the sheet resistance is 12.4Ω and the specific contact resistivity is $24.3\Omega\mu m^2$. The calculated resistance between the two electrodes is 19.3Ω including the contact resistance. For comparison, we replace the contact electrode by a boundary electrode around the contact region and perform the simulation again. The calculated total resistance is 10.8Ω for this case. An 8.5Ω difference is due to the effect of contact resistance. If a circuit layout has a large number of such contacts, neglecting them in circuit extraction may cause an incorrect prediction of circuit performance in the next step of verification.



$\rho_c = 24.3\,\Omega\,\mu m^2$

$R_s = 12.4\,\Omega$

Figure 2.8 An example layout with a contact electrode.

Example 4:

In this example, we use a square coaxial transmission line [25] shown in Figure 2.9 to evaluate the calculated accuracy. The exact resistance of this structure is known to be $1.287095144\,\Omega$ if the sheet resistance is $1\,\Omega$. Table 2.7 shows the percentage errors and the calculation times for different numbers of grid points chosen in FEMRC. The calculation time includes grid generation, triangulation, matrix solving and other setup times. It is observed that increasing the number of triangular elements improves the calculated accuracy, just as theoretically predicted. But the calculation time is significantly increased if a large number of grid points are used in the calculation. For the practical application, however, a trade-off between the calculated accuracy and the computation time should be made in order to save the computer resources.

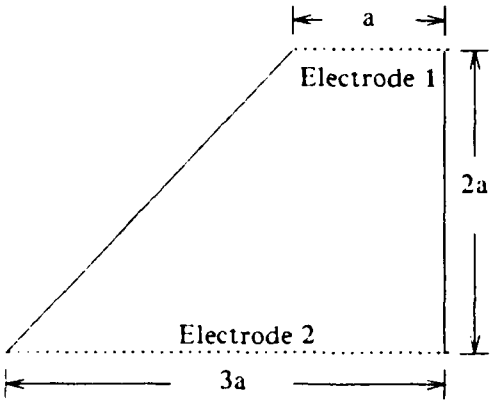


Figure 2.9 One eighth of cross section of a square coaxial transmission line.

Table 2.7 Runtimes and percentage errors of resistance calculations for the conductor shown in Figure 2.9 with different triangulations.

case	number of nodes	number of triangles	runtime† (sec)	error (%)
(1)	174	293	36	3.82
(2)	679	1256	213.6	2.96
(3)	1270	2345	505.7	1.24

†SUN 3/160

2.6. Conclusions

In this chapter, a two-dimensional capacitance and resistance calculation program FEMRC based on FEM has been developed. In the capacitor model, there are no restrictions on the number of dielectrics and conductors, and the shape of the conductors and the dielectric interfaces. In the resistor model, any arbitrarily shaped, multielectrode and multiple resistivity

resistor can be simulated. In addition, contact resistance is modeled by the Helmholtz equation, in which quasi-three-dimensional current flow is taken into consideration. Experience with FEMRC shows that it is a practical useful tool in VLSI performance verification.

At the present time, FEMRC is not incorporated into the layout extraction program HPEX. However, it generates the reference values for analytical capacitance and resistance models employed in HPEX. Details for the analytical models for capacitance and resistance will be discussed in the next chapter.

CHAPTER 3.

RESISTOR AND CAPACITOR MODELS IN HPEX

3.1. Introduction

Several methods have been developed to calculate distributed resistances and capacitances of interconnection lines in a circuit extraction program. In general, these methods can be classified into the following three categories: (1) numerical methods [33-34, 24], (2) table-look-up methods [3], and (3) analytical formulas [19-20, 35].

Numerical methods, as described in the previous chapter, are used to compute capacitances and resistances by discretizing the field equation in the space or the surface of the conductor of interest. Since the accuracy of the computation strongly depends on the number of grid points, a large number of grid points generally needs to be generated in order to obtain a reasonably satisfactory result. This in turn implies that a large number of linear equations have to be solved. Therefore, numerical methods often require a large amount of computer memory and time. Hence, although they are able to model the real shapes of conductors easily and very accurately, numerical methods are impractical for the majority of parasitic extractions of VLSI circuit layouts from the computational standpoint. One practical application of the numerical methods is that they can be used to model a small number of irregular layouts, while allowing more regular shaped conductors to be handled by other simpler models.

The table-look-up method and analytical formula approach are simpler in comparison to the numerical model. Table-look-up methods are fairly efficient, but constructing tables for all possible layout configurations would be time-consuming. Therefore, analytical formulas are employed in our resistance and capacitance computations for simplicity and efficiency. Some

accuracy may be lost if analytical formulas are adopted. However, by carefully fitting two- or even three-dimensional numerical data, analytical formulas still produce results with good accuracy. Furthermore, since only Manhattan layouts are allowed as the input to our circuit extractor, derivation of analytical formulas with reasonable accuracy is much easier. Our goal is, therefore, to find accurate analytical formulas for resistances and capacitances in actual circuit layouts and use them in the circuit extractor. In this manner, interconnect parasitics can be extracted in a reasonable amount of CPU time, as well as being modeled with reasonable accuracy.

3.2. Resistor Models Used in the Extractor

3.2.1. Long, straight interconnect conductors

Since our extractor handles only Manhattan circuit layouts, simple formulas have been employed to compute the resistances of nets. For long, straight conductor lines, the well-known one-dimensional resistance formula given below is used.

$$R = R_{sh} \frac{length}{width} \tag{3.1}$$

where R_{sh} is the sheet resistance. This means that the value of resistance in a long conductor line is the sheet resistance times the total number of squares in the line. When rectangular conductors of the same material with various widths are cascaded, as shown in Figure 3.1, the total resistance is approximated by the following equation:

$$R = R_{sh} \sum_{i}^{N} \left| \frac{L_i}{W_i} \right| \tag{3.2}$$

This is only an approximation, since the current crowding effect around some corners of rectangles has been neglected. The accuracy of this approximation is hard to predict, since it varies with the actual layout implementation. For example, the actual resistance of the layout shown

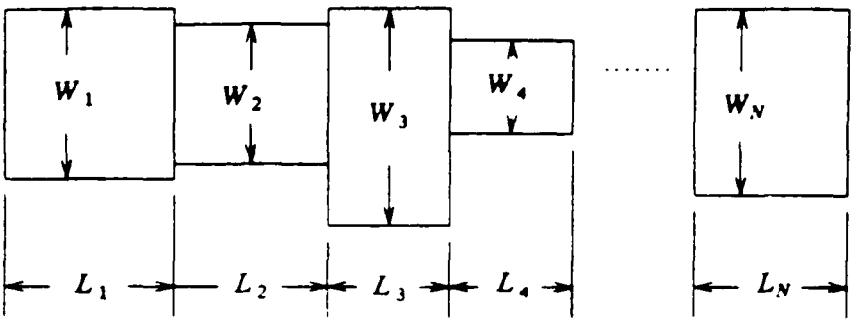


Figure 3.1 Cascaded rectangular conductors.

in Figure 3.2 is $2.12R_{sh}$, while the approximate resistance is $2R_{sh}$. The percentage error is [illegible] in this case. The percentage error will be reduced if we change squares into rectangles by m[illegible] ing two electrode sides outwardly. However, if we only slide the left square up until the length of the common segment is four times shorter than the original one, the percentage er[illegible] by applying Equation (3.2) will be increased. Despite the difficulty in controlling the accura[illegible] Equation (3.2) is still strongly favored by a circuit extractor because of its simplicity.

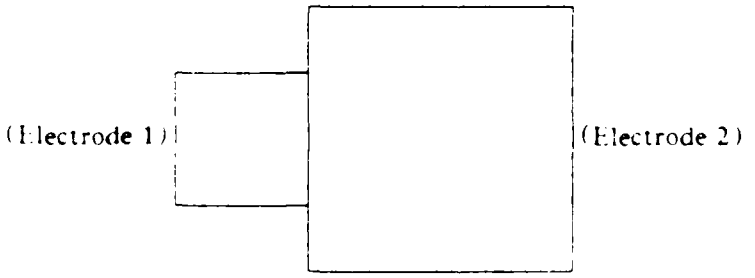


Figure 3.2 Two cascaded conductors.

3.2.2. Right-angle bend in the conductors

An important shape that we have taken into account is the right-angle bend, as shown in Figure 3.3 in the conductors. If the corner rectangle is a square, the resistance of the corner rectangle is less than that computed by using the center line as the length due to current crowding. The two-dimensional analytic formula we use to approximate the resistance of the corner rectangle is [36]:

$$R_c = \left| \frac{1}{a} - \frac{2}{\pi} \ln \left| \frac{4a}{a^2+1} \right| + \left| \frac{a^2-1}{\pi a} \right| \cos^{-1} \left| \frac{a^2-1}{a^2+1} \right| \right| R_{sh} \quad . \tag{3.3}$$

where a is the ratio of wide-to-narrow widths of the corner rectangle. In Figure 3.3, $a = \frac{w_1}{w_2}$ with the assumption that $w_1 \geq w_2$. It should be noted that if the corner rectangle is a square, i.e., $a = 1$ the resistance R_c is $0.559 R_{sh}$ instead of R_{sh} computed by Equation (2.1).

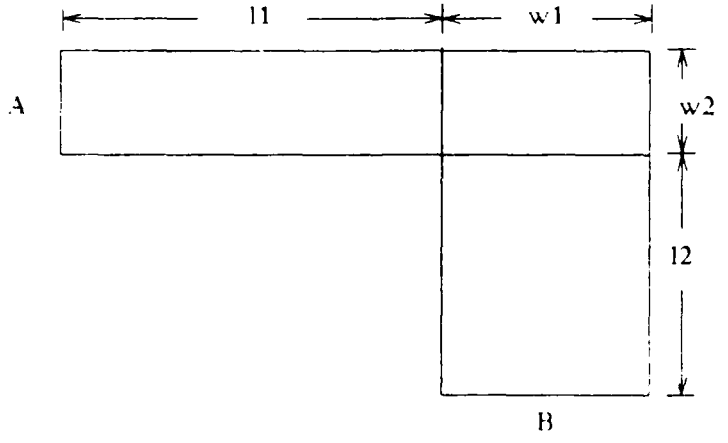


Figure 3.3 A right-angle bend.

3.2.3. Irregular shapes of conductors

For other more complicated shapes in interconnects, the Laplace equation must be solved in order to determine their resistances. However, this is not done in our extractor in order to save CPU time. Instead, we use a rectangle partitioning algorithm mentioned in the previous chapter to estimate resistances of some irregular shapes. For instance, Figure 3.4(a) shows an actual interconnect layout. After partitioning one rectangle, a new node number must be inserted, and the result is shown in Figure 3.4(b). The resistance of the rectangle, which abuts the other three rectangles, is approximated by half of the resistance determined by the above-mentioned formula. If the Laplace equation is solved for this configuration, we have the more accurate result shown in Figure 3.4(c). For comparison, this circuit can be transformed into one equivalent to Figure 3.4(b), as shown in Figure 3.4(d). Although our assumption is crude, the error is still tolerable

3.2.4. Highly irregular shapes of conductors

If a small portion of a highly irregular shaped conductor requires the high accuracy of the resistance calculation, the only feasible method is to solve the 2-dimensional Laplace equation by the numerical program FEMRC described in Chapter 2. At present this program is not included in our extractor. The designer has to interactively simulate the critical part of the layout, then insert the extracted resistance network back to the circuit extractor output. The combined circuit description is then ready for the circuit simulation. Due to the required user interaction and the fairly heavy computation by FEMRC, this approach might not be very practical.

FEMRC is also able to simulate two-dimensional capacitances by adopting that option in running the program. Therefore, FEMRC can also be used to verify the 2-d analytical capacitance formulas which will be discussed in Section 3.3.

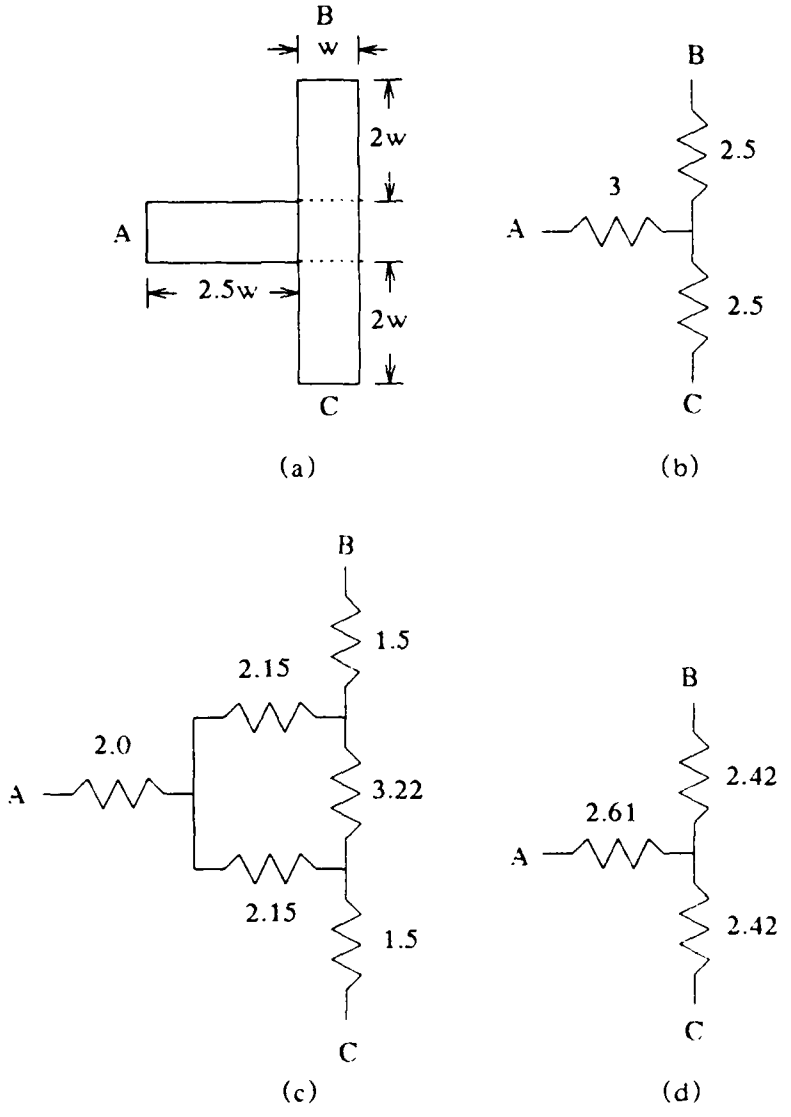


Figure 3.4 A resistance calculation example: (a) A conductor layout, (b) An equivalent resistance network calculated by our method, (c) A resistance network calculated by FEMRC, and (d) An equivalent network for the network shown in (c).

3.2.5. Contact resistance

As MOS technology scales down into the submicron regime, the series resistances contributed by the source-drain and polysilicon contacts may become important. The contact resistance R_c is defined as the resistance between two different materials in the contact region. The most commonly used contact resistance is computed by the following equation.

$$R_c = R_{sh} \frac{L_c}{W \tanh(L/L_c)} . \tag{3.4}$$

where L is the length of window contact, W is the width of window contact which is perpendicular to the current flow, and L_c is the transfer length and has the following form:

$$L_c = \left(\frac{\rho_c}{R_{sh}} \right)^{1/2} . \tag{3.5}$$

where ρ_c is the specific contact resistivity, the physical parameter which governs the interfacial contact resistance between two different materials.

Equation (3.4) is derived from the one-dimensional model and is usually used in a transmission line tap test structure [30] to determine ρ_c. However, in this model it is assumed that the specific contact resistivities for diffusion-metal and polysilicon-metal contacts are already given. If the direction of the current flow is known, Equation (3.4) then can be applied to estimate the contact resistance in the circuit extraction. Unfortunately, this information will not be known until the simulation is performed. Therefore, the following zero-dimensional analytic equation is adopted to compute the contact resistance.

$$R_c = \frac{\rho_c}{A} , \tag{3.6}$$

where A is the contact area. This equation assumes that the potentials at the contacted materials are constant, and the current density entering the contact window is uniform. It can be shown that Equation (3.6) is a limiting case of Equation (3.4) when $L_c >> L$. If a more accurate estimate for the contact resistance is desirable, the numerical model embedded in FEMRC or

even a three-dimensional model should be employed. However, in our extractor only Equation (3.6) is applied.

3.3. Capacitor Models Used in the Extractor

In calculating capacitance of the interconnects, basically, two types of capacitances are considered: the self-capacitance (line-to-ground capacitance) and the coupling capacitance. Coupling capacitances can be further categorized into two types: (1) the parallel-edge capacitance for two conductors on the same mask, and (2) the overlap capacitance for two conductors on two different masks.

3.3.1. Empirical formulas for line-to-ground capacitances

To accurately predict the VLSI performance during the preliminary circuit design phase, accurate estimation of parasitic capacitances associated with interconnects is inevitable. In this section, we will concentrate on the derivation of one of the most important parasitic capacitances, i.e., line-to-substrate capacitance or self-capacitance. Although, generally, two- or three-dimensional numerical calculations using Green's function [37] or the finite-element method as described in Chapter 2 are able to predict the line-to-substrate capacitance of a conductor in a homogeneous medium or on several layers of the media, they are prohibitively expensive to be incorporated into CAD programs. A simple, accurate and closed-form expression to approximate the line-to-substrate capacitances is thus necessary and critical in this situation.

Several closed-form formulations of a single conductor in a homogeneous medium have been proposed [38-40]. However, they are either computationally complicated or not always accurate. In this section, a simple and accurate formula for the line-to-substrate capacitance of a single conductor in a homogeneous medium is described. In the formulation the two-

dimensional effects, e.g., fringing fields and the thickness of a conductor, are taken into account. Furthermore, an attempt is made to include the effect of the slant sidewalls of a conductor into the formulation. To confirm the accuracy of the reported formula, programs such as CAP2D or FEMRC are used to generate the reference values. To evaluate the slant-sidewall effect on the reported formula, examples with various combinations of design parameters are examined. Finally, a simple modification of the approximate formula suitable for two layers of media is discussed.

3.3.1.1. Formulation

The capacitance of a conductor over a substrate shown in Figure 3.5 is determined by the thickness t, the width w of a conductor, the distance h from the ground plane (the substrate usually taken as the ground), and the angle α of the slant sidewalls. Intuitively, this line-to-substrate capacitance is composed of two parts, one with $t=0$ and the other resulting from the sidewalls of a conductor. If the capacitance with $t=0$ and the capacitance due to the sidewalls of a conductor are correctly modeled, then the line-to-substrate capacitance is simply given by

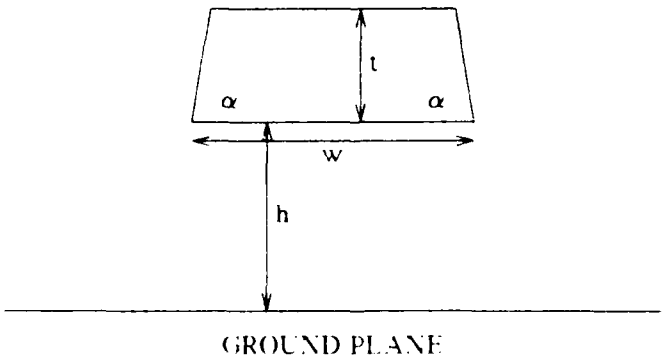


Figure 3.5 A conductor over ground plane.

$$C = C_p + C_s \tag{3.7}$$

where C_p is the capacitance of infinitesimally thin parallel plates ($t=0$), including the fringing field effect, and C_s is the capacitance due to the sidewalls of a conductor

For the capacitance of the very thin parallel plates ($t=0$), including the fringing field effect, the following formula is first considered [41]

$$C_p = \frac{\epsilon w}{h} + \left[1 + \frac{2h}{\pi w}\left[1 + \ln\left[\frac{\pi w}{h}\right]\right]\right], \quad \frac{w}{h} >> 1 \tag{3.8}$$

The first term represents the parallel-plate capacitance while the second term is the fringing-field capacitance. However, this formula is only valid when the width w is much greater than the distance h from the ground plane. The errors are more pronounced when the width w is comparable to the distance h, the usual case for VLSI circuits. Hence, other formulas which have been developed for the microstrip using a modified conformal transformation method (for $t=0$) are employed [42].

$$C_p = \epsilon\left[\frac{w}{h} + 1.393 + 0.667 \ln\left[\frac{w}{h} + 1.444\right]\right], \quad w \geq h \tag{3.9}$$

$$C_p = \frac{2\pi\epsilon}{\ln\left[\frac{8h}{w} + \frac{w}{4h}\right]}, \quad w < h \tag{3.10}$$

The fringing-field capacitance C_f of a single conductor with $t=0$ is given by

$$C_f = \frac{C_t - C_p}{2} \tag{3.11}$$

The capacitance C_p is the parallel-plate capacitance given by the following simple equation:

$$C_p = \epsilon\frac{w}{h} \tag{3.12}$$

As for the sidewall capacitance of a single conductor, the following formula which models two conducting plates at a certain angle is used [43].

$$C_u = \frac{\epsilon \ln\left|l_2/l_1\right|}{\theta} \tag{3.13}$$

where l_1, l_2 and θ are the lengths and the angle as shown in Figure 3.6. Since the fringing-field effects at the ends are neglected in the above formula, the computed capacitance always underestimates the actual one. But this underestimation will be compensated by the capacitance of thin parallel plates (t=0) mentioned when summing up the total line-to-substrate capacitance. Therefore, the total line-to-substrate capacitance per unit length of a single conductor is given by

$$C = \epsilon\frac{w}{h} + \epsilon\left[1.393 + 0.667 ln\left(\frac{w}{h} + 1.444\right) + \frac{2}{\pi - \alpha}\ln\left(1 + \frac{t}{h}\right)\right], \quad w \geq h \tag{3.14}$$

$$C = \epsilon\frac{w}{h} + \epsilon\left[\frac{2\pi}{\ln\left(\frac{8h}{w} + \frac{w}{4h}\right)} - \frac{w}{h} + \frac{2}{\pi - \alpha}\ln\left(1 + \frac{t}{h}\right)\right], \quad w < h \tag{3.15}$$

It is noted that the factor 2 in the sidewall capacitance term accounts for the two sidewalls of a single conductor. Also the capacitance is reduced to the capacitance of thin parallel plates C_u when t=0, in accordance with the physical interpretation.

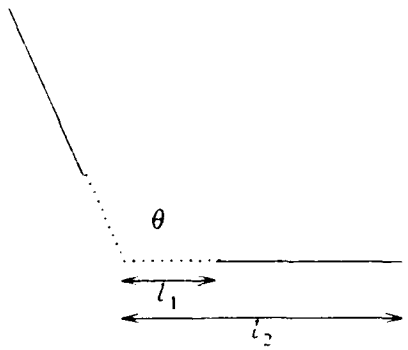


Figure 3.6 Two plates with a certain angle.

3.3.1.2. Comparison and discussion

In order to compare the accuracy of the various proposed approximations, capacitance values obtained by two-dimensional numerical computations using Green's function were chosen as the reference values. Capacitances normalized with respect to the parallel-plate capacitance obtained by numerical calculations and our approximation with $\alpha = 90^{\circ}$ are depicted in Figure 3.7 for $t/h = 0.1$, 1, and 10, respectively. For $t/h = 0.1$ and 1 our approximated values are within 3% of the reference values except for $w/h << 1$. This is because the formula for the capacitance of thin parallel plates used in our approximation is highly inaccurate when $w/h << 1$. In VLSI circuits, however, this is seldom the case. For $t/h = 10$ the observed errors are within 6% of the reference values again except for $w/h << 1$ for the same reason as before.

In Figures 3.8, 3.9 and 3.10, the percentage error with $\alpha = 90^{\circ}$ is plotted for $t/h = 0.1$, $t/h = 1$ and $t/h = 10$, respectively, for Elmasry's approximation [39], the cylindrical approximation [40], and our approximation. These figures show that over a wide range of design parameters, i.e., the thickness and the width of a conductor or the distance of a conductor from the substrate, the results seem to be much better when our formulas are used to compute the capacitance of a conductor with the sidewall angle $\alpha = 90^{\circ}$. In order to study the capacitance of a conductor with slant sidewalls, a two-dimensional numerical program which includes the slant-sidewall effect was used to generate the reference values. Table 3.1 lists the percentage errors for conductors with a sidewall angle $\alpha = 70^{\circ}$, and the results show a general agreement between the approximate values and the reference values. Also indicated in Table 3.1 are the percentage errors between calculations neglecting the slant-sidewall effect in our approximate formulas ($\alpha = 90^{\circ}$) and calculations by the two-dimensional program with $\alpha = 70^{\circ}$. The comparisons show that except for $t/h = 0.1$ the percentage error is greater for our approximate formulas where the sidewall angle α is not taken into consideration. Therefore, in order to maintain the accuracy of our proposed formulas, it is important to take the slant-sidewall effect into account.

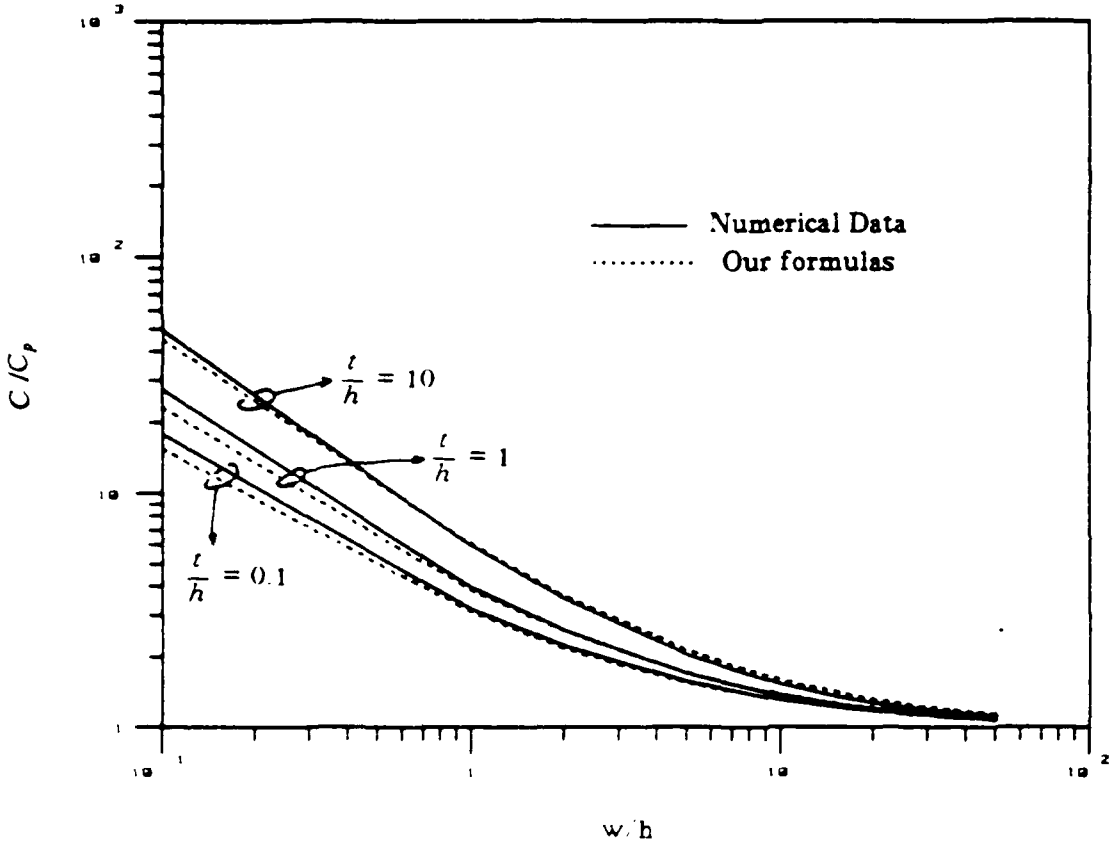


Figure 3.7 Normalized capacitances computed by numerical method and our formulas for t/h = 0.1, 1, and 10.

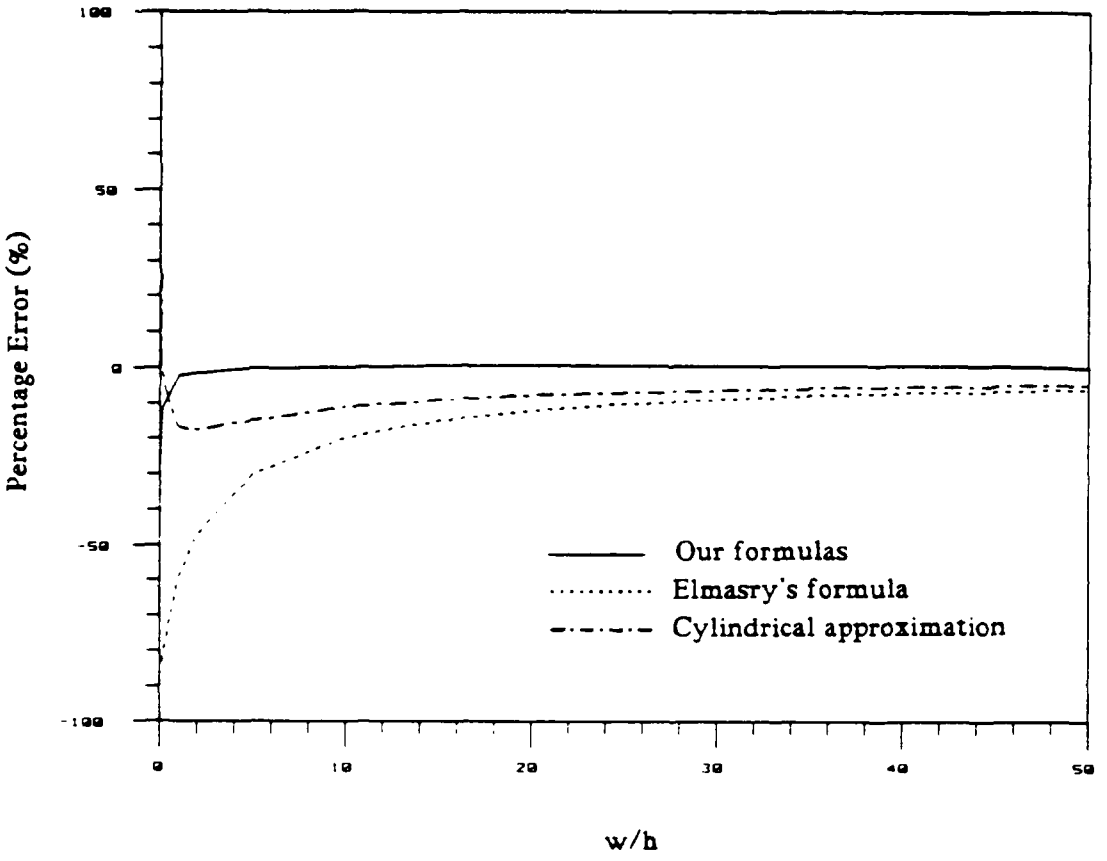


Figure 3.8 Comparison of percentage errors for t/h = 0.1.

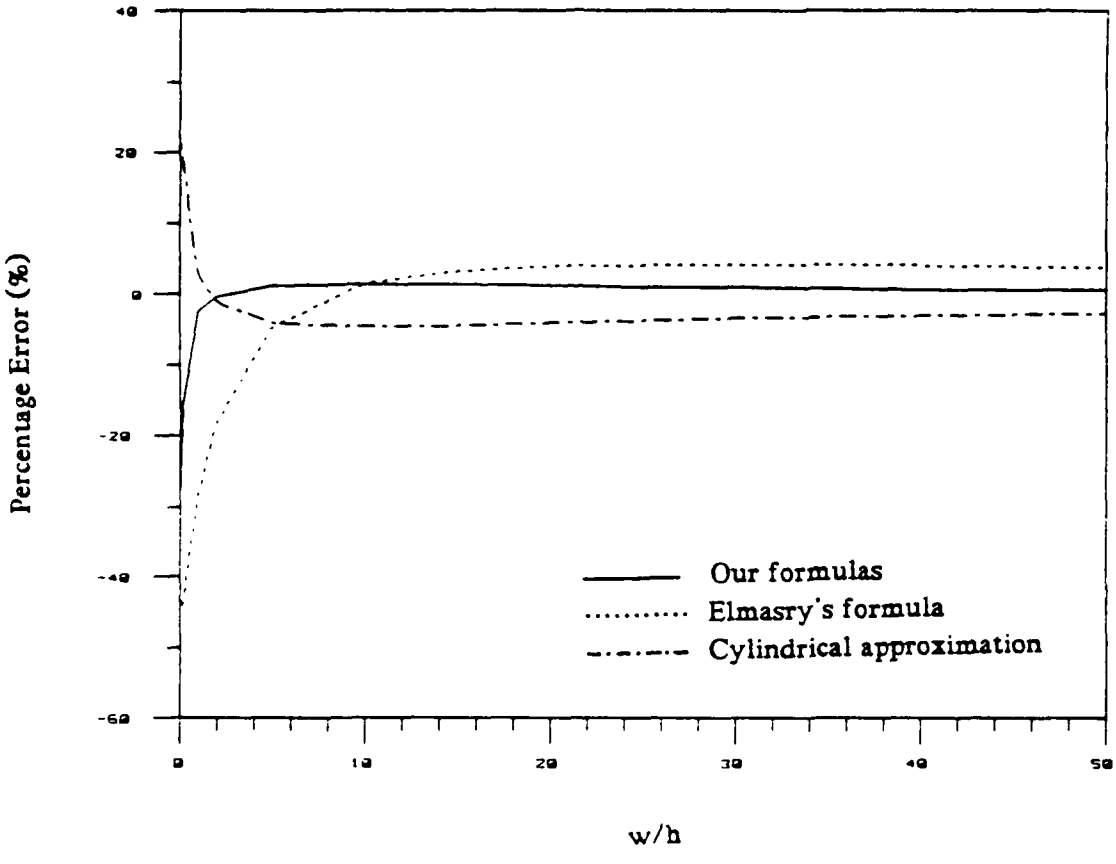


Figure 3.9 Comparison of percentage errors for t/h = 1.

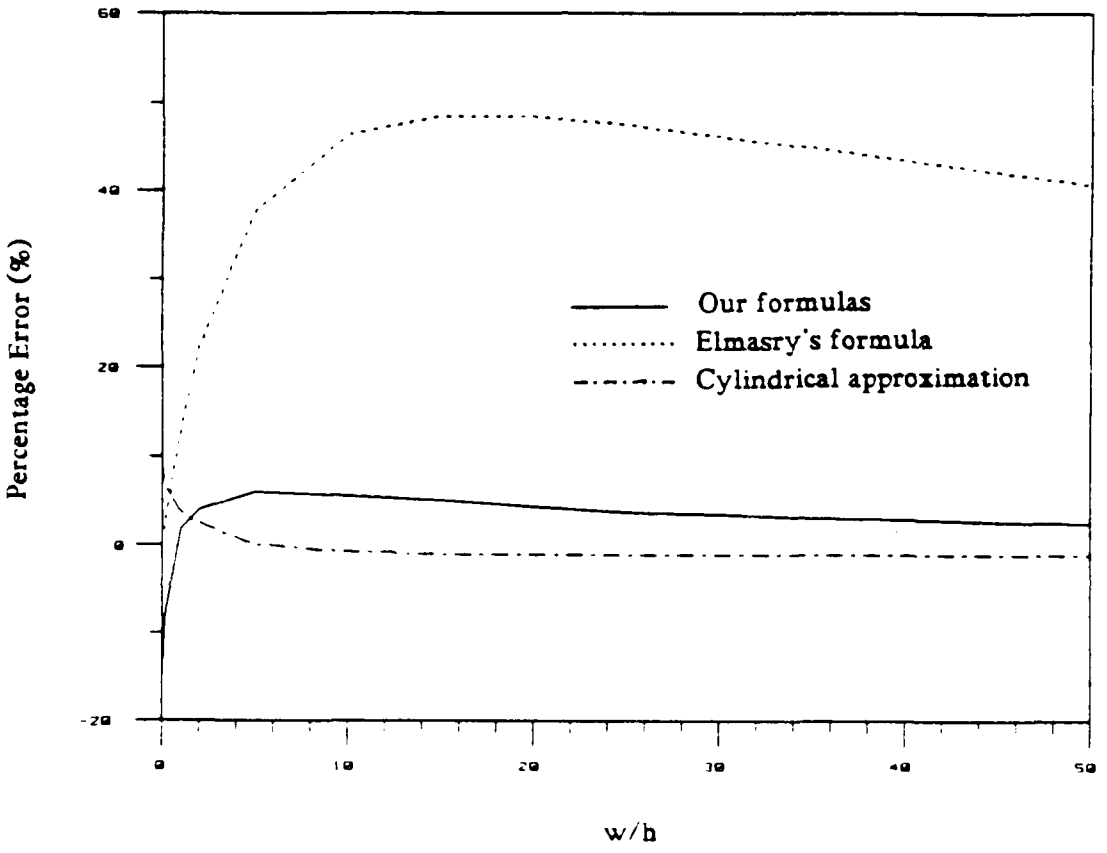


Figure 3.10 Comparison of percentage errors for t/h = 10.

Table 3.1 Percentage errors of our self-capacitance formulas for a conductor with a sidewall angle $\alpha=70^{\circ}$.

Percentage Error ($\alpha=70^{\circ}$)					
w/h	t/h = 10	t/h = 5	t/h = 3	t/h = 1	t/h = 0.1
1	-	-	-	0.33 (4.67)†	-1.82 (-1.11)
2	-	-	-	0.97 (4.24)	-1.34 (-0.83)
5	-	6.48 (11.14)	4.73 (8.48)	1.70 (3.65)	-0.25 (0.08)
10	6.92 (10.71)	5.12 (8.02)	3.88 (6.18)	1.63 (2.81)	0.29 (0.45)

†percentage error for $\alpha=90^{\circ}$.

In the previous discussion we have assumed a single dielectric. In an actual configuration, however, the conductor is situated on top of the oxide (SiO_2) and capped with a passivation layer, e.g., SiN as shown in Figure 3.11. The difference between dielectric constants should also be taken into account in the approximate formulas to maintain accuracy. By using the similar treatment as [40], weighting factors for the "parallel-plate" term and the "fringing" term are chosen. For two layers of dielectric media the line-to-substrate capacitance is thus given by

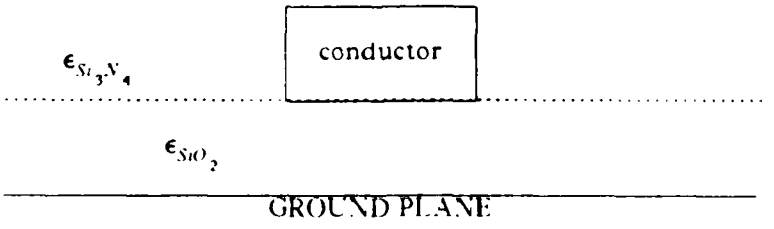


Figure 3.11 A conductor surrounded by two different dielectrics.

$$C = \epsilon_{ox} \left| parallel-plate\ term + \frac{\epsilon_{avg}}{\epsilon_{ox}} \left(fringing-field\ term \right) \right|. \tag{3.16}$$

where $\epsilon_{avg} = \frac{1}{2}\left(\epsilon_{ox} + \epsilon_{SiN}\right)$. To examine the error produced by the above equation, reference values were generated by a two-dimensional program assuming that the thickness of the passivation layer is infinite. The percentage errors of different combinations of w/h and t/h for α=90° and 70° are listed in Tables 3.2 and 3.3, respectively. The dielectric constants used are ϵ_{ox} = 3.9, ϵ_{SiN} = 7.5. Note that the errors are within 6% of the reference values. This again shows that our approximations are reasonable, even for a conductor in the two layers of dielectric media.

Table 3.2 Percentage errors of Eq. (3.16) for α=90° as compared to numerical calculations.

Percentage Error (%) (α=90°)			
w/h	t/h = 10	t/h = 1	t/h = 0.1
1	-5.22	-3.96	-4.27
2	-2.26	-2.30	-2.68
10	-2.98	-1.70	-1.55

Table 3.3 Percentage errors of Eq. (3.16) for α=70° as compared to numerical calculations.

Percentage Error (%) (α=70°)			
w/h	t/h = 10	t/h = 1	t/h = 0.1
1	-	-1.16	1.96
2	-	0.95	2.58
10	-4.26	0.06	1.57

3.3.2. Coupling capacitance

We shall now turn to the coupling capacitance calculation. For the case as shown in Figure 3.12, the following simple formula for parallel-edge capacitance [35] is used.

$$C_{12} = \epsilon \left[0.03 \left(\frac{w}{h} \right) + 0.83 \left(\frac{t}{h} \right) - 0.07 \left(\frac{t}{h} \right)^{0.222} \right] \left(\frac{s}{h} \right)^{-1.34} . \tag{3.17}$$

where w is the width of the conductor, t is the thickness of the conductor, h is the height of the conductor-to-ground plane and s is the separation between two conductors. The percentage error of the above formula is within 10% of the referenced value resulting from 2-d calculation.

As for the case in which two parallel conductors are on different mask levels as shown in Figure 3.13, the following formula for the overlap capacitance derived in [27] is used.

$$C_{12} = \epsilon_1 \frac{d}{t_1} + \epsilon_1 \frac{0.5\pi}{\ln\left[1+\gamma+\left[\gamma\left[\gamma+2 \right] \right]^{\frac{1}{2}} \right]} + \epsilon_{avg} \frac{0.5\pi}{\ln\left[1+\beta+\left[\beta\left[\beta+2 \right] \right]^{\frac{1}{2}} \right]} . \tag{3.18}$$

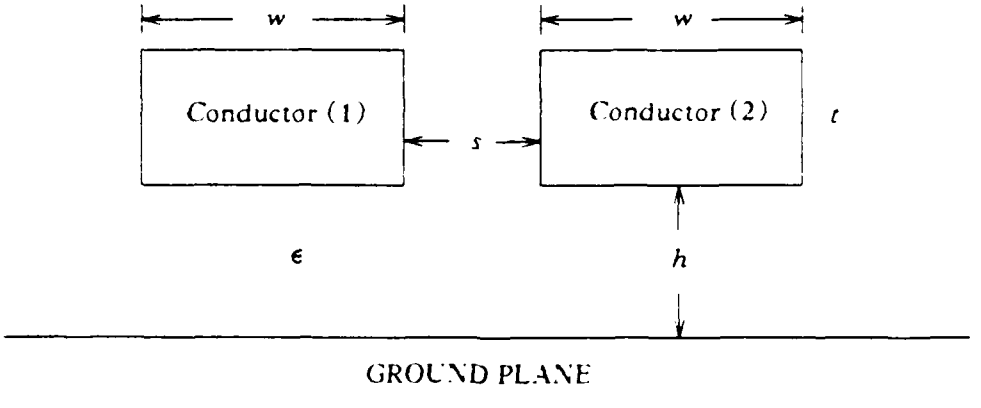
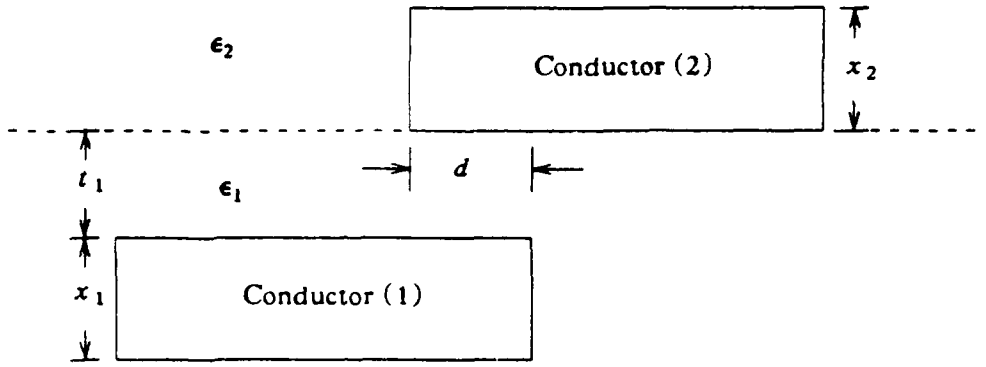


Figure 3.12 Two conductors at the same layer over ground plane.



GROUND PLANE

Figure 3.13 Two conductors with overlapping.

where $\gamma = \frac{4t_1}{x_1}$, $\beta = \frac{4t_1}{x_2}$, and $\epsilon_{avg} = (\epsilon_1+\epsilon_2)/2$. This formula accounts for different dielectric constants by using a weighting factor.

Finally, the formula for cross-over coupling capacitance as derived in [27] is employed in our extractor, which is given by

$$C_{12} = \epsilon\left[0.0034\left(\frac{w}{t}\right)^{3.243} - 5.624\left(\frac{h}{t}\right)^{0.0175} + \left(\frac{s}{t}\right)^{-0.354}\left[6.74\left(\frac{w}{t}\right)^{1.118} + 6.25\left(\frac{h}{t}\right)^{0.171}\right]\right]. \qquad (3.19)$$

This formula is obtained by fitting data calculated by a 3-dimensional numerical program. In order to reduce variables in curve fitting, two assumptions are made. First, thickness and widths are taken as the same for two conductors. Secondly, the length is five times that of the

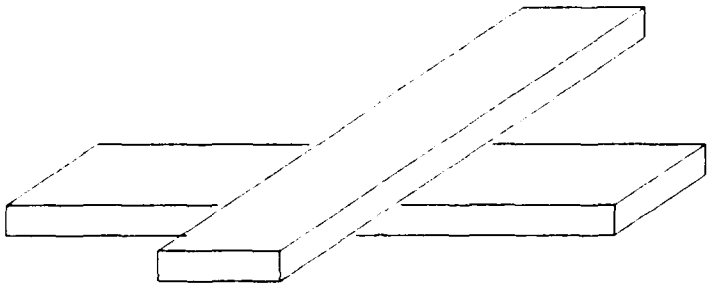
width. This assumption is a reasonable one because most of the fringing field is close to the overlapped region between two conductors. The nominal percentage error is 10% as compared to the numerical calculation. However, if the thickness and width of both conductors are different, this percentage error might increase.

3.4. Lumped RC Circuits for Interconnect Modeling

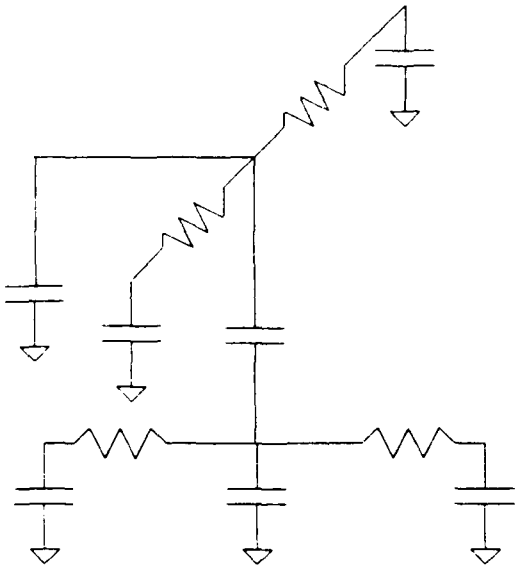
Once resistance and capacitance values are determined, a lumped circuit model is used to approximate the behavior of distributed *RC* lines. In [44], it has been concluded that π-circuit models give satisfactory results in the approximation of a distributed *RC* line. Therefore, a π-circuit model is chosen to approximate every branch in the interconnect. Furthermore, in order to compromise between the number of nodes created and the accuracy of modeling, a resistance threshold R_π is used to select a one- or a two-section π-lumped circuit in modeling distributed interconnects. However, for the insertion of coupling capacitance, a T-like circuit model is chosen for simplicity. Figure 3.14 gives an example showing how we use the lumped circuit model to approximate distributed RC lines which include the coupling capacitance.

3.5. Conclusions

In this chapter, the resistance and capacitance computation models used in HPEX are developed. In modeling an interconnection line, analytical formulas obtained by fitting the numerical data are first employed to compute the resistance and capacitance values. These values are then distributed into two sections of a lumped circuit. Analytical formulas, instead of more accurate numerical methods, are adopted in HPEX to significantly reduce the computation time. By carefully fitting the analytical formulas, the produced errors can be controlled within the tolerance limit.



(a)



(b)

Figure 3.14 Lumped *RC* model for cross-over interconnects.

The same fitting technique can be applied to derive the closed-form formulas for coupling capacitance calculation. The accuracy of the fitting formulas are somewhat reduced because most of the coupling capacitances are three-dimensional in nature. Furthermore, some accuracy may be lost due to the fact that self- and coupling capacitances are separately computed in these models. In reality, both self- and coupling capacitances are strongly coupled together and cannot be separated. However, these computation models are able to reduce the computer

resources by predicting the first-order parasitic effects. If a higher-order accuracy is required, a numerical program such as FEMRC should be included in HPEX. But for real circuits with a large number of parasitics, the incorporation of FEMRC into HPEX may not be practical.

CHAPTER 4.

DATA STRUCTURE AND BASIC ALGORITHMS IN HPEX

4.1. Technology Assumptions

Technology assumptions embedded in HPEX can be described as follows:

(1) MOS technology : Either NMOS or CMOS circuit layouts are assumed.

(2) Manhattan style layout : Only isothetic (or rectilinear) polygons are allowed as input to HPEX. All boundary segments of isothetic polygons are parallel to x-axis or y-axis. Furthermore, all isothetic polygons are split into a equivalent set of rectangles stored in the internal data structure of HPEX.

(3) No closed loop transistors: Only simple or serpentine shape transistors can be specified in the input file.

(4) One- or two-layer metal process : No contact cuts are allowed between the second layer metal and polysilicon or diffusion layer. The second layer metal is electrically connected to the polysilicon or diffusion conductors only through the first layer metal.

(5) Planar interconnect process : There is no step coverage in the interconnect conductors. Although this assumption simplifies resistance and capacitance calculation, it is somewhat invalid because most of present semiconductor technologies still use the conventional interconnect process with step coverages. The planar interconnect technology will substantially complicate the fabrication process. However, the planar interconnect process will become the choice of technology due to the requirement for multi-level interconnects and

high packing density in VLSI circuits. Some of advanced GaAs circuit technologies have already demonstrated the advantages of using the planar interconnect technology.

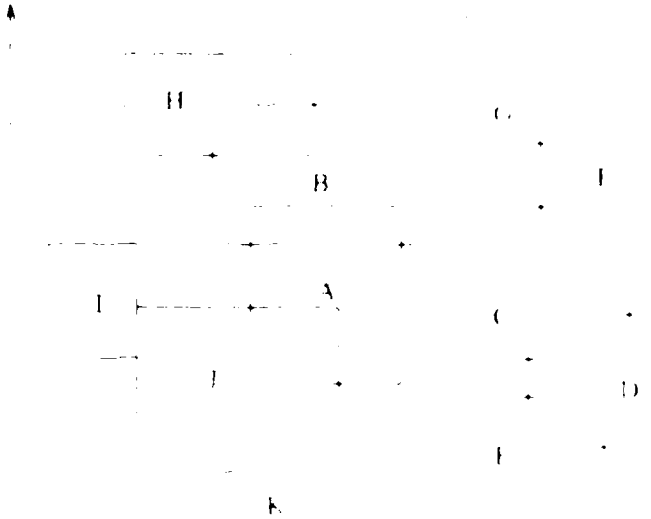
4.2. Geometrical Data Structure and Algorithms

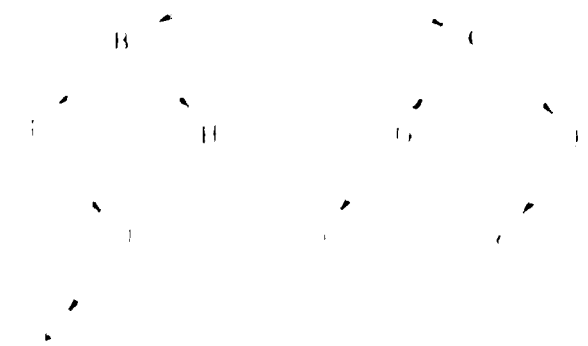
In HPEX, a layout is represented as a collection of rectangles. All geometric manipulations, such as boolean mask operations, are based on rectangles. In general, a large number of rectangles are needed to describe the layout of a typical VLSI circuit. Therefore, from the extractor performance point of view, to design an efficient rectangle data structure for HPEX is very crucial. A linked list of rectangles, though simple to implement, is not efficient enough for region queries. Region queries, frequently used in HPEX, are defined as operations of finding all rectangles intersecting a specified region. The geometrical data structure employed in HPEX is called a 4-d binary search tree [45-46]. The basic concept behind a 4-d binary search tree is based on a divide-and-conquer technique. Each node of the tree corresponds to a rectangle in the layout. At the root of the 4-d binary tree, the space is split in half by the left edge of the rectangle corresponding to the root node. At the next level, the remaining space is cut in half by the bottom edge of corresponding rectangles. By a similar procedure the right edge and then the top edge is used to divide space in half. Figure 4.1 shows a layout example consisting of 11 rectangles and its corresponding 4-d binary tree.

The Pascal data structure for 4-d binary trees used in HPEX is defined as follows:

```
rectptr = ^rect;

rect   = record
         key    : array[0..3] of real;
                  (* four keys:
                  key[0] = xmin
                  key[1] = ymin
                  key[2] = -xmax
                  key[3] = -ymax *)
         mask   : mask level; (* Mask level rectangle resides on *
         lson,  (* left son in tree structure *)
         rson   : rectptr; (* right son in tree structure *
         ndisc  : integer; (* discriminator of successor
```





1 Diffusion
2 Poly
3 Polysilicon
4 Contact Cut
5 Metal
6 Contact Cut 2
7 Metal 2
8 Buried Contact
9 P-plus
10 Overglassing

In a 4-d binary search tree each tree node is defined in terms of a rectangle which has key[0], key[1], key[2], and key[3]. Four keys of a rectangle are defined as follows:

key[0] =
key[1] =
key[2] =
key[3] =

key[0], key[1], key[2], key[3] are the lower left and upper right coordinates of a rectangle. Each node of a 4-d binary search tree contains two pointers to sons called *lson* and *rson*. DISC(P) of a node P same as "ndisc" in our record) is defined in the range [0..3]. We compare key[DISC(P)] and key[DISC(P)+2] and determine the direction to the next node in the tree. In a 4-d binary search tree, the discriminator starts at 0 at the root and is incremented by 1 for each successive level of penetration of the tree until the value 3 is reached, and then it is reset to 0. For any node Q in the left subtree of P it is true that key[DISC(P)] > key[DISC(P)] for node Q in the right subtree of P, key[DISC(P)] < key[DISC(P)]. For example key[0] is the key to determine if the right subtree of the root is the desired subtree at the root which has the discriminator value 0. As the tree is deeper, the discriminator value is increased by 1 and the appropriate key values are applied to determine discrimination.

if DISC(P) = 0
then if key[0] < key[0] then go to left subtree of P
else if key[0] > key[0] then go to right subtree of P
else if DISC(P) = 1
then if key[1] < key[1] then go to left subtree of P
else if key[1] > key[1] then go to right subtree of P
else if DISC(P) = 2

In the input phase of HPEX, 4-d binary search trees are first built according to each mask level. Then various searches are performed based on these trees, either identifying transistor channels or finding all electrically connected rectangles. In general, two basic algorithms are needed: insertion and intersection queries. An insertion algorithm is used to build as well as update 4-d binary search trees, while an intersection search algorithm is to find all rectangles in a tree which are intersected by a given region. The insertion algorithm used in HPEX is described as follows.

ALGORITHM 4.1 4DINSERT

Input: a node P and a 4-d binary search tree.
Output: a new 4-d binary search tree with node P inserted.

I1 [Check for empty tree]
If ROOT = nil then set ROOT := P, DISC(P) := 0,
rson(P) := nil, lson(P) := nil, set rson(P) as thread and return
Otherwise Q := ROOT

I2 [Compare]
If $key_P[i] = key_Q[i]$ for $0 <= i <= 3$
(i.e. the nodes are equal) then return; otherwise,
if
(1) $key_P[DISC(Q)] > key_Q[DISC(Q)]$;
or (2) $key_P[DISC(Q)] = key_Q[DISC(Q)]$;
$key_P[(DISC(Q)+1) mod 4] > key_Q[(DISC(Q)+1) mod 4]$;
or (3) $key_P[DISC(Q)] = key_Q[DISC(Q)]$;
$key_P[(DISC(Q)+1) mod 4] = key_Q[(DISC(Q)+1) mod 4]$;
$key_P[(DISC(Q)+2) mod 4] > key_Q[(DISC(Q)+2) mod 4]$;
or (4) $key_P[DISC(Q)] = key_Q[DISC(Q)]$;
$key_P[(DISC(Q)+1) mod 4] = key_Q[(DISC(Q)+1) mod 4]$;
$key_P[(DISC(Q)+2) mod 4] = key_Q[(DISC(Q)+2) mod 4]$;
$key_P[(DISC(Q)+3) mod 4] > key_Q[(DISC(Q)+3) mod 4]$;
then set son(Q) := rson(Q)
else set son(Q) := lson(Q).
If son(Q) = nil, then go to I4

I3 [Move down]
Set Q := son(Q) and go to I2.

I4 [Insert new node in tree]
Set son(Q) := P, rson(P) := nil, lson(P) := nil and return.

In general, a built tree is not balanced in height if a random order of rectangles is used as an input. By employing the above algorithm, Figure 4.2 illustrates a 4-d binary search tree

corresponding to the layout shown in Figure 4.1. This tree structure is different from that shown in Figure 4.1 because in the actual implementation *key*[2] and *key*[3] are defined as negative of the right and upper bounds of a rectangle. It should be noted that for applying the above algorithm a different insertion order of rectangles may produce a different tree structure. As an example, Figure 4.3 shows a different 4-d binary tree corresponding to the same layout described in Figure 4.2. Since a built tree structure varies with the input sequence of rectangles, we may have a purely directed path as the worst case in applying 4DINSERT.

Another algorithm to build a 4-d binary search tree attempts to balance the height of the tree by some preprocessing steps at each level of the tree structure. This alternative first reads in all rectangles and stores them in a linked list. This linked list is then recursively sorted by the respective key, and broken into approximately two halves by a median node. Given a

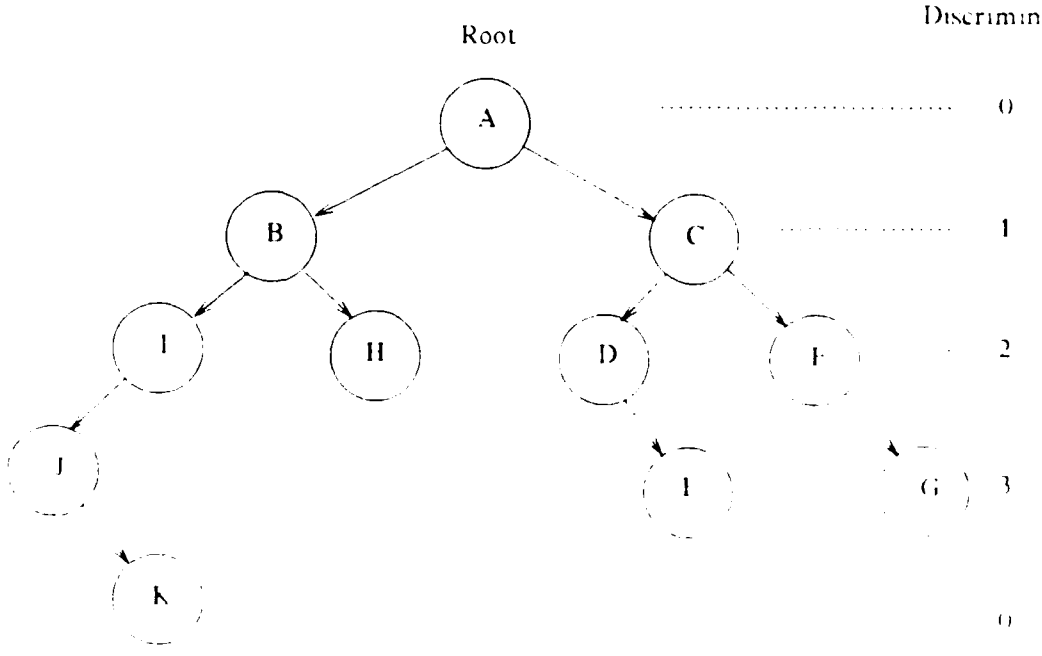


Figure 4.2 4-d binary search tree constructed by 4DINSERT from the layout shown in Figure 4.1

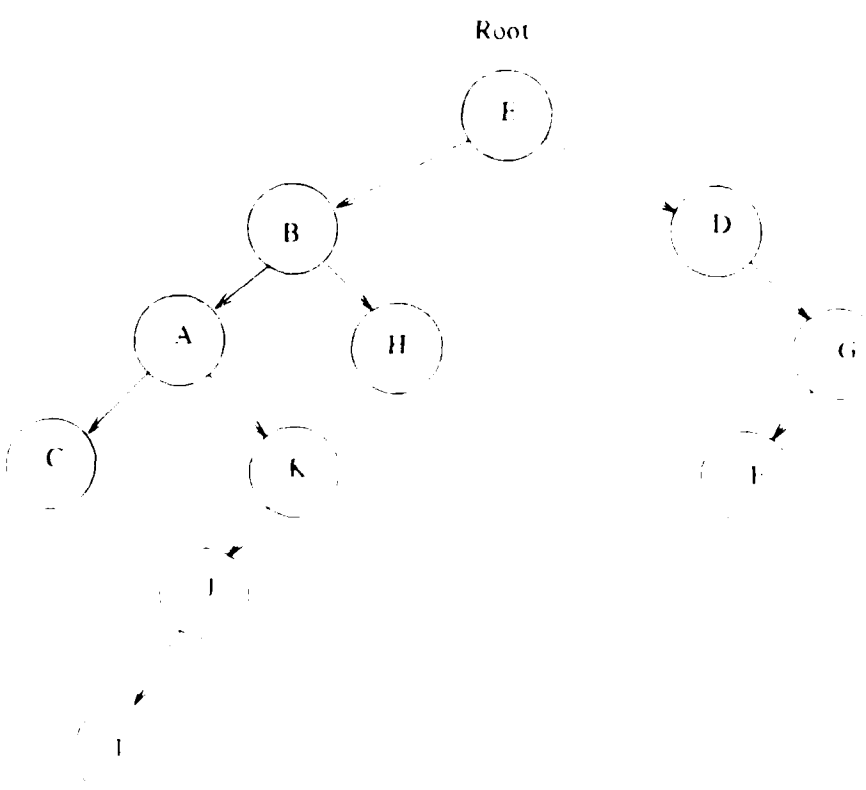


Figure 4.3.4 A binary search tree with different insertion order

linked list A of n rectangles sorted by the x-co- in ascending order the *median node* of A is defined as follows

$$\text{median node} = \left\lfloor \frac{n-1}{2} \right\rfloor + 1^{\text{th}} \text{ node} \quad \text{if } n \text{ is odd number}$$

$$\text{median node} = \frac{n}{2} \text{ node} \quad \text{if } n \text{ is even number}$$

Details of this recursive algorithm are given in the next

ALGORITHM 4.2 4DBUILD(A,j,P)

Input: a linked list of nodes A, discriminator j,
    and root node P
Output: a balanced tree rooted at P

B1 [Check for empty set]
  If A is nil then return

B2 [Find median]
  Set P = j median node of A

B3 [Split set]
  Set AL = {Q ∈ A : Q is j less than P}
  AR = {Q ∈ A : Q is j greater than P}

B4 [Recur]
  Set j = NEXTDISC(j)
  4DBUILD(AL, j, son(P))
  4DBUILD(AR, j, son(P))

[illegible] linked [illegible] rectangles [illegible] a particular mask [illegible] 4DBUILD(B,0,ROOT) [illegible] balanced tree. For example, Figure 4.4 [illegible] a balanced 4-d tree [illegible] Figure 4.3. [illegible] be pointed out that the representation [illegible] each [illegible]

[illegible] 4DINSERT [illegible]

ALGORITHM 4.3 4DSEARCH(Q,P,j)

[illegible]

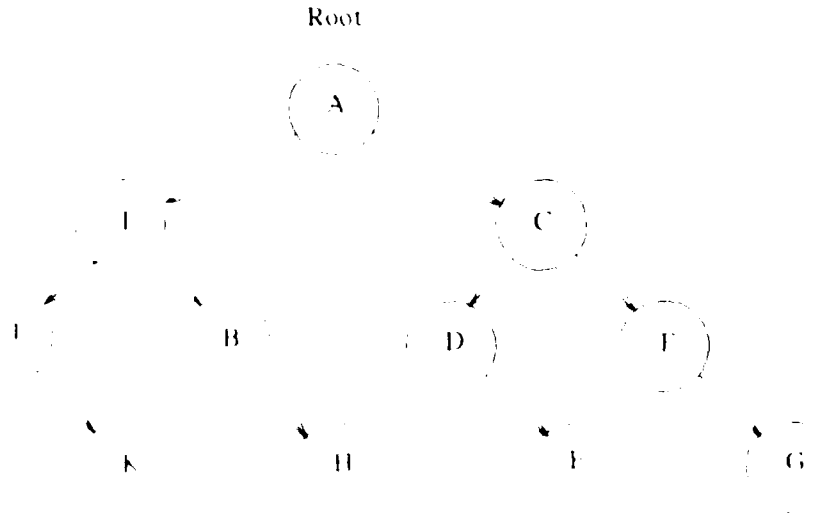


Figure 4.4 A balanced 4-d binary search tree

4.3. Circuit Netlist Data Structure

4.3.1. MOS transistor record

Since our circuit extraction is oriented to the MOS technology, the record for MOS transistors must be defined in order to store the extracted information about the transistors. The transistor record is defined as follows:

```
tranptr = ^transistor;

transistor = record
        ttype : nmostype; (* type of the transistor *)
        width,length : real; (* width and length of the transistor *)
        source,drain : adjptr;
        gate : pdaptr;
        dnode,snode,gnode : integer; (* node numbers for drain, source and gate *)
        gcap : real; (* WLC_ox *)
        dcap : real, (* capacitance associated with drain node *)
        scap : real; (* capacitance associated with source node *)
        drainnet,sourcenet,gatenet : netptr;
        next : tranptr;
       end;
```

In the above record, most of the parameters are self-explanatory. The *source*, *drain* and *gate* pointers are heads of lists of rectangles corresponding to the source, drain and gate regions, respectively. This information is very important when we estimate the equivalent channel length, width, source and drain areas of the extracted MOS devices. In order to easily fetch three terminal nets of a particular transistor, *sourcenet*, *drainnet* and *gatenet* are defined as pointers to source, drain and gate nets, respectively. It is also noted that the drain, source and gate node numbers are kept in the integers *dnode*, *snode* and *gnode*, and need to be updated during the extraction process.

4.3.2. Interconnection records

The net record which represents all information in modeling interconnect RC is extremely important in circuit extraction and is defined as follows:

```
netptr    = ^net;
net         = record
               ntype , nettype;
               namenode : ionodeptr; (* character name *)
               totalres,totalcap : real;
               tranconset : tranlinkptr;
               reduction : boolean; (* need node reduction? *)
               bound : array [0..3] of real;
               nextnet : netptr;
               number : integer; (* net number *)
               geometry : gmyptr;
               branch : branptr;
               resistor,selfcap : crptr; (* lists of resistors and capacitors *)
               couplingnet : cpnetlinkptr; (* coupling nets *)
              end;
```

The pointer *tranconset* points to a list of connected transistors by the net. This information is useful when we need to traverse the circuit schematic. A list of rectangles which constitutes the net is stored in the *geometry* pointer. These rectangles are processed through the *RC* model module to generate interconnect parasitic resistances and capacitances. If the net is connected to more than two transistors, then branches constituting the net are pointed by the *branch* pointer. Another parameter used is *bound* , which defines the boundary of a net. Before calculating the coupling capacitance between two nets, *bound* is used to determine whether the coupling is too small to calculate. This avoids computation of the coupling capacitances which have a negligible effect on circuit performance.

The basic unit in interconnect *RC* modeling for HPEX is called *branch*. A *branch* is defined as a set of electrically connected rectangles, in which there is no side branch for current flow. For each branch, two resistors and three capacitors are used to approximate an *RC* distributed line. The branch record in HPEX is given as follows:

```
branptr    = ^branch

branch      = record
               bound : array [0..3] of real;
               node1,node2,nodem : integer; (* end and middle node numbers *)
               resistor1,resistor2 : crptr;
               node1cap,node2cap,nodemcap : crptr;
               gmyconnect : adjptr;
               coupcap : cpconptr;
```

```
    next : branptr;
        end;
```

The array *bound* also represents the bounding box of the branch. In coupling capacitance calculation, if two nets are close enough, then the bounding boxes of two branches from each net are compared to filter out the coupling effect due to a large distance. This process proceeds until all branches have been compared. This hierarchical comparison for the bounding boxes of nets and branches will substantially reduce the number of rectangles required for comparison during coupling capacitance calculation.

4.4. Layout Resizing Algorithms

Because of the important role played by chip layout feature sizes in circuit performance, actual dimensions of devices and interconnects should be correctly estimated. If the CIF input represents a mask layout, then the difference between the mask layout and the actual dimension on the physical layout resulting from process variations should be taken into account during circuit extraction. The fabricated regions are usually different from the regions in the mask layout by a uniform width around the periphery of the regions. For example, the diffusion regions are larger than the mask layout due to the inherent lateral diffusion of ions during the fabrication process. Nevertheless, the polysilicon regions might be smaller than the designed mask because of the process bias. This width of contraction or expansion becomes significant in determining the critical feature size of the circuit layout as the VLSI chip becomes larger and denser. Thus a capability to resize the regions of a VLSI layout before circuit extraction is important for the layout verification.

Given the regions covered by a set of rectangles, several approaches have been proposed to contract or to expand these regions. Typically, a layout resizing procedure can be described as follows. First, the outlines (or boundaries) of the disjoint regions are detected. Then the true boundaries of the resized regions could be formed by contracting (or expanding) the boundary

segments inwardly (or outwardly) through parallel displacements. If the final output is represented by an equivalent set of rectangles, a decomposition algorithm which transforms polygons into rectangles is required. Therefore, two different data structures are necessary to support the resizing algorithm: (1) a rectangle data structure and (2) a polygon data structure. An algorithm employed to maintain these data structures during structural transformation could be complicated and difficult to implement. Hence a simple and efficient resizing algorithm based only on one rectangle data structure is proposed to contract or expand the input circuit layout. This resizing algorithm is implemented in the extractor HPEX.

In the remaining part of this section, important definitions will first be given in order to describe the layout resizing problem. Some of the previous approaches will then be reviewed, followed by a discussion of our proposed resizing algorithm. Finally, other potential applications of the proposed algorithm will be described.

4.4.1. Definitions

Since a rectangle is chosen as a primitive in our internal data structure, the following definitions are mainly aimed at the relationships between rectangles.

Definition 4.1 Isothetic Rectangle

An *isothetic rectangle* is a rectangle with its edges parallel to either x-axis or y-axis. In the remaining part of this chapter, the term 'rectangle' represents isothetic rectangle for brevity.

Definition 4.2 Overlap

Two rectangles are overlapped if they have a common area.

Definition 4.3 Abut in the x (or y) Direction

Two rectangles are abutted in the x (or y) direction if they have a common line segment which is parallel to y-axis (or x-axis).

Definition 4.4 Equivalence

A set of rectangles $\mathbf{R}_1$ is equivalent to another set of rectangles $\mathbf{R}_2$ if they cover the same regions.

Definition 4.5 Expansion (or Positive Offsetting) by Width d

Let $\mathbf{S}$ be a rectangular region. The expanded region of $\mathbf{S}$ by width d is defined as follows

$$\mathbf{E}_d\mathbf{S} = \{u \in \mathbf{R}^2 \mid \forall v \in \mathbf{S}, |u - v| < d\} \tag{4.1}$$

where $\mathbf{E}_d$ is an expansion operator which expands a region by width d.

For example, Figure 4.5 shows a rectangle (solid line) and its expanded region (dotted line). It can be observed that, by applying the above definition, the expanded regions have circular arcs of radius d, which inevitably complicate the resizing algorithm and its supporting data structure. As a result, in order to simplify the resizing algorithm and data structure, the following loose definition for expansion of a rectangle is used hereafter.

Definition 4.6 Parallel Expansion by Width d

Let $\mathbf{S} = \{(x,y) \in \mathbf{R}^2 \mid x_1 \leq x \leq x_2 \text{ and } y_1 \leq y \leq y_2\}$ be a rectangle in the plane $\mathbf{R}^2$. The parallel expanded region of $\mathbf{S}$ by width d is defined as follows:

$$\mathbf{E}_d^{\parallel}\mathbf{S} = \{(x,y) \in \mathbf{R}^2 \mid x_1 - d \leq x \leq x_2 + d \text{ and } y_1 - d \leq y \leq y_2 + d\}, \tag{4.2}$$

where $\mathbf{E}_d^{\parallel}$ is a parallel expansion operator that expands a region outwardly by a parallel movement of boundary segments for a distance d.

Figure 4.6 illustrates a rectangle and its parallel expansion. According to this definition, rectangles remain rectangles after expansion. The same is true for polygons. However, for polygons the parallel expansion is not as simple as that for rectangles, since there might be an intersection of boundary segments after expansion as shown in Figure 4.7. Therefore, finding intersections between boundary segments after expansion and removing false boundary segments become two main steps in the polygon expansion algorithm. As for rectangles, we can easily find their expanded regions without worrying about intersections between boundary segments. This is one of the reasons that we develop the resizing algorithm based on a rectangle data structure.

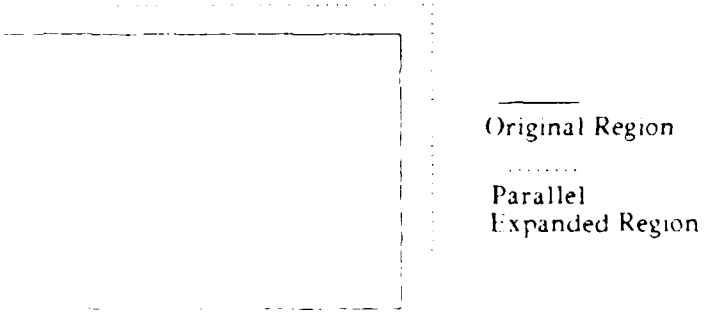


Figure 4.6 Layout parallel expansion

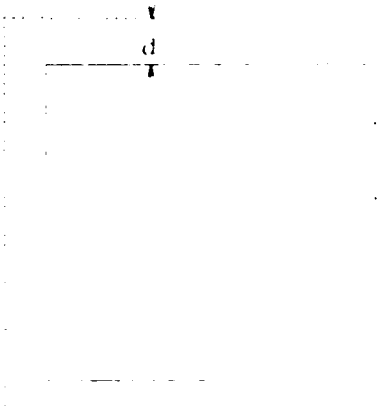


Figure 4.7 Layout expansion with intersection of

Definition 4.7 Complement Operator

For any region $\mathbf{R}$, a complement operator $\hat{C}$ is defined as follows

$$\hat{C}\mathbf{R} = \{(x, y) \in \mathbf{R}^2 \mid (x, y) \notin \mathbf{R}\} \tag{4.3}$$

Since the contraction of a rectangle $\mathbf{S}$ can be defined as the complement of the expansion's complement, namely $C_j^i\mathbf{S} = \hat{C}(E_j^i(\hat{C}\mathbf{S}))$, where C_j^i and $\hat{C}$ are parallel contraction operator and complement operator respectively, we will not give a separate definition about rectangle contraction here. It follows immediately from the above definitions that the following boolean operations are correct

$$E_j^i(\mathbf{A} \bigcup \mathbf{B}) = E_j^i\mathbf{A} \bigcup E_j^i\mathbf{B} \tag{4.4}$$

$$C_j^i(\mathbf{A} \bigcup \mathbf{B}) \supset C_j^i\mathbf{A} \bigcup C_j^i\mathbf{B} \tag{4.5}$$

The first property indicates that the expansion of a union of the rectangles is equivalent to the union of the expansions of each individual rectangles. From the electrical point of view, given a set of rectangles in the same mask level, if two rectangles are electrically connected, then they

4.4 Previous approaches to layout resizing problems

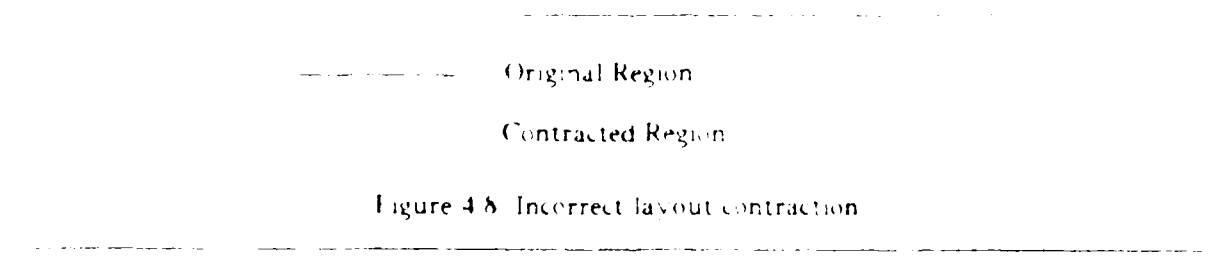


Figure 4.8 Incorrect layout contraction

4.4.2.1. Outline procedure

Given a set $\mathbf{R}$ of N rectangles $R_1, \ldots, R_N$, the outline procedure is to find the outline of $I = \bigcup R_i$. The set I may consist of one or more disjoint connected components and the outline of I consists of disjoint cycles constructed by vertical and horizontal segments.

The standard algorithm and its detailed implementation of the outline procedure for N given rectangles can be found in the text authored by Preparata and Shamos [47]. The algorithm has two main phases and can be briefly described as follows. In the first phase, the scan line technique supported by the segment tree $\mathbf{T}$ [47] is employed to find the set $\mathbf{V}$ of vertical segments of the outline. Once the set $\mathbf{V}$ has been obtained, the second phase is to link these vertical segments by means of horizontal segments to form the outline of I.

Another outline algorithm has been proposed by Komatsu and Suzuki [48]. The main concept behind their algorithm is first to find disjoint connected components (each component consists of a list of rectangles) of rectangles allocated to subchips of a chip area. Then the rectangles that belong to the same polygon are collected and the outline segments are extracted. By repeating this process, the outline procedure is carried out. The process of extracting the contour from a set of rectangles is similar to that mentioned in the first algorithm except that the set of horizontal edges instead of the vertical edges are derived first.

4.4.2.2. Resizing procedure

After the outlines of a given set of rectangles have been detected, we can proceed to resize these regions. This is equivalent to resizing a set of polygon regions. The typical resizing algorithm can be found in [49] and is summarized as follows:

XY Method

[1] Scan the input region vertices and sort them in the ascending y-coordinate order. As shown in Figure 4.9(a), the polygon is then split into banded regions according to this list of sorted vertices.

[2] Let d be the expansion distance. For each banded region, the distance from a vertical side A to its neighboring side B is determined. If it is less than $2d$, then the area between A and B is added to the figure and two banded regions are merged horizontally (Figure 4.9(b)).

[3] Expand the output regions from Step 2 by d in the x direction since no intersection of sides will occur (Figure 4.9(c)).

[4] Repeat the previous steps for the y direction.

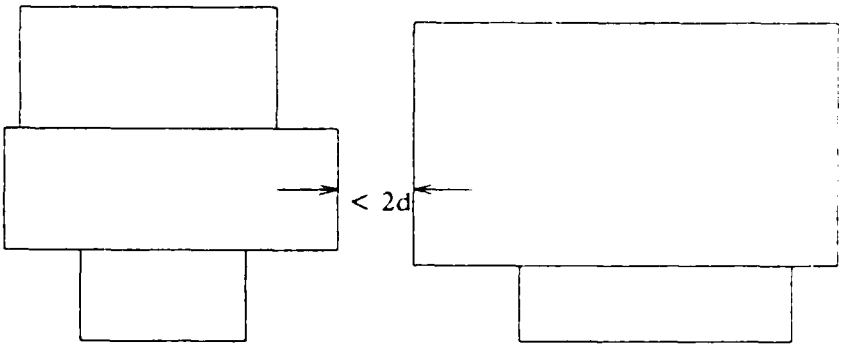
Step 2 can be done by scanning the vertices from large to small y values. In each position of the scanline, the horizontal test is performed to find banded regions which should be merged after the expansion. Also note that this method resizes the rectilinear regions with x-axis and y-axis directions separately.

4.4.3. Our approach to layout resizing problem

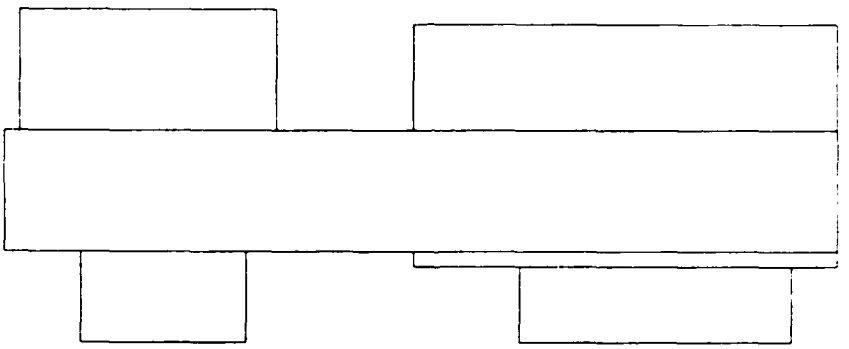
For the purpose of efficient implementation of the resizing algorithms, we develop two different algorithms (expansion and contraction) in our approach instead of applying the concept of region complement.

4.4.3.1. Expansion algorithm

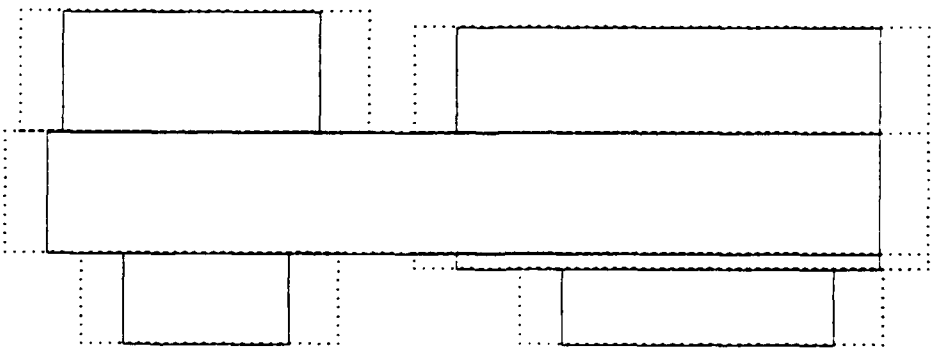
As mentioned before, there is no discontinuity problem for the region after a rectangle-by-rectangle expansion. Therefore, in the expansion problem we can first expand every



(a) X direction banded regions



(b) Horizontal merge



(c) X direction expansion

Figure 4.9 XY method.

rectangle, and then remove all the overlapped regions. The algorithm we propose is straightforward and can be described as follows:

ALGORITHM 4.4 Scanline Method for Layout Expansion

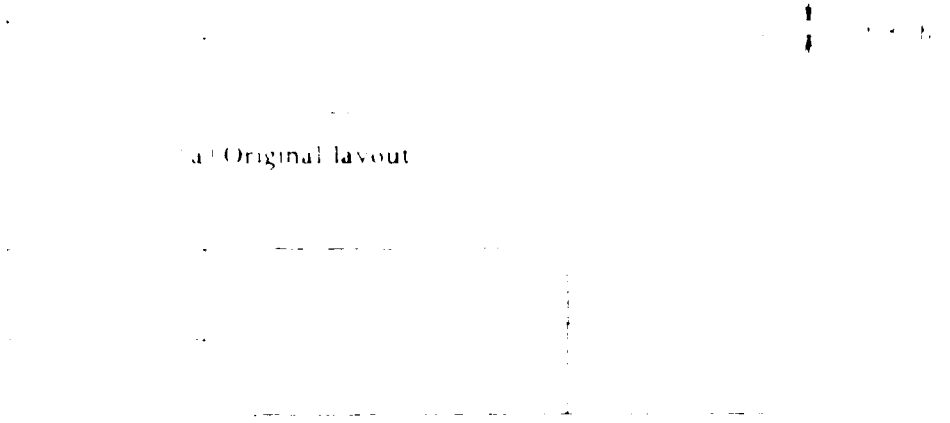
Input: a list of boxes **B** in one mask level, the shrinkage distance d
and the feature size threshold h
Output: a list of nonoverlapping expanded boxes $\mathbf{B}_f$

begin
1. Expand every box in **B** by a distance d.
2. Sort **B** by top edge of box in descending order
3. Use scanline sweeping in -y direction to
transform **B** into $\mathbf{B}_{ff}$ which represents
a set of maximum horizontal strips
4. Sort $\mathbf{B}_{ff}$ by right edge of box in descending order
5. Set scanline at right edge of layer, sweep in the -x direction
to vertically merge boxes with y direction feature size less
than predefined threshold h into contracted set $\mathbf{B}_f$
6. return($\mathbf{B}_f$).
end.

Step 2 in the above algorithm is to produce a unique layout representation with boxes adjacent to each other in the X-direction. It is noted that in this step the "scanning painting algorithm" similar to [50] is used to remove all overlaps for each mask level in order to correctly perform circuit extraction. The purpose of step 5 is to construct the correct dimensions for all boxes to be used in the resistance model for circuit extraction. Figure 4.10 shows an example for step 5.

4.4.3.2. Contraction algorithm

Rectangle-by-rectangle contraction for the layout resizing might create the electrical discontinuity problem. Therefore, traditional contraction algorithm merges all boxes into disjoint polygons before going to the actual layout contraction. However, in order to simplify the algorithm and its supporting data structure, we propose a contraction algorithm which is based on a novel Y-X scanline method together with a rectangle data structure in our extractor. This algorithm is described below.



(a) Original layout

(b) After vertical merge

Figure 4.10 A vertical merge example.

ALGORITHM 4.5 Scanline Method for Layout Contraction

Input: a list of boxes **B** in one mask level, the shrinkage distance d, and the feature size threshold h.
Output: a list of contracted boxes $\mathbf{B}_c$.

begin
1. Sort **B** by top edge of box in descending order.
2. Use scanline sweeping in -y direction to transform **B** into $\mathbf{B}_{yy}$ which represents a set of maximum horizontal strips.
3. Shrink every box in $\mathbf{B}_{yy}$ by distance d.
4. Set scanline at top position of layer again, sweep layer in -y direction to insert missing boxes due to layer contraction.
5. Sort $\mathbf{B}_{yy}$ by right edge of box in descending order.
6. Set scanline at right edge of layer, sweep in the -x direction to vertically merge boxes with y direction feature size less than pre-defined threshold h into contracted set $\mathbf{B}_c$.
7. return($\mathbf{B}_c$).
end.

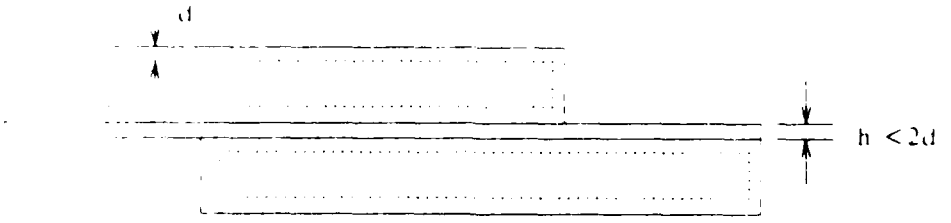
It is noted that shrinking boxes is performed after a set of maximum horizontal strips are

[illegible] However, for the [illegible] we expand [illegible] **B**

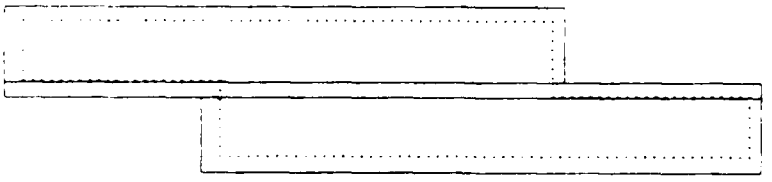
[illegible] algorithm can apply to most VLSI layouts it might not work [illegible] layout. The example is shown in Figure 4.11, in which if the feature [illegible] twice the resizing distance, the above algorithm will produce the undesirable [illegible] In order to make the algorithm more robust, a modified algorithm called [illegible] scanline method is developed and given as follows.

ALGORITHM 4.6 Modified Scanline Method for Layout Contraction

Input: a list of boxes **B** in one mask level, the shrinkage distance d and the feature size threshold h
Output: a list of contracted boxes $\mathbf{B}_c$.



(a) Incorrect Contraction



(b) Correct Contraction

Figure 4.11 An example to show incorrect contraction by scanline algorithm.

```
begin
  1 Sort B by right edge of box in descending order
  2 Use scanline sweeping in -x direction to
    transform B into B_x which represents
    a set of maximum vertical strips
  3 Shrink every box in B_x by distance d in y
    direction
  4 Sort B_x by top edge of box in descending order
  5 Use scanline sweeping in -y direction to
    transform B_x into B_y which represents
    a set of maximum horizontal strips
  6 Shrink every box in B_y by distance d in x
    direction
  7 Set scanline at right edge of layer, sweep in the -x direction
    to vertically merge boxes with y direction feature size less
    than predefined threshold h into contracted set B_c
  8 return(B_c).
end
```

The first three steps in the above algorithm are oriented at contracting the regions in the y direction, while the next three steps are for contracting in the x direction. It can be easily shown that after applying the above algorithm the regions satisfy the definition of parallel contraction. A step by step example of this algorithm is illustrated in Figure 4.12.

4.4.4. Other applications

4.4.4.1. Pattern data preparation for E-beam lithography

The writing speed is one of the most important requirements in the electron beam direct writing used in the VLSI fabrication process. The vector-scan method has high potential for the high-speed pattern generation because it does not have to scan regions where no pattern exists. In the vector-scan system, however, complicated procedures are required to prepare data for writing. This is because the patterns which can be written in such systems must be represented in primitive shapes, such as rectangles. Therefore, the designed shapes have to be preprocessed in order to produce suitable data for the vector-scan system. An important step in preparing

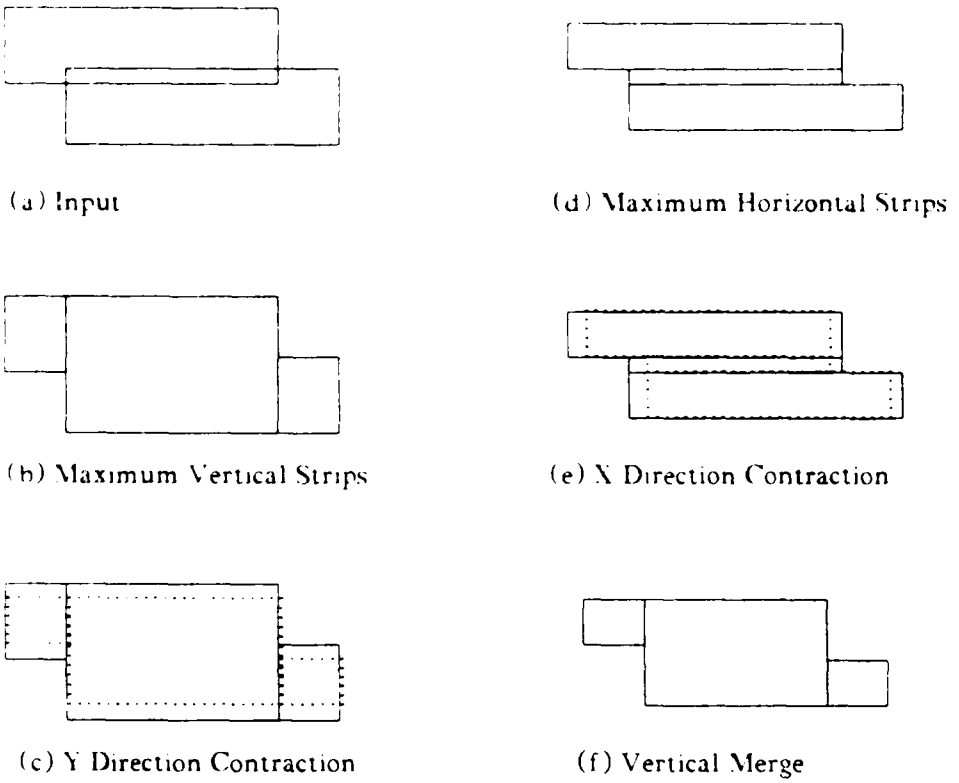


Figure 4.12 Modified scanline algorithm for contraction.

pattern data is to eliminate overlap regions. This step can easily be achieved by the algorithms described in the previous section.

4.4.4.2. Compensation for proximity effect

The proximity effect in electron-beam lithography is the phenomenon that the pattern data receive a nonuniform distribution of exposure due to forward-scattered and backward-scattered electrons within both resist and substrate regions. It will produce undesirable shapes after fabrication, and probably alter the circuit performance. One of the methods to compensate for this effect is to resize the input patterns according to adjacent geometries. For a shape with

denser adjacent patterns, the edges of this shape facing these patterns should be inwardly displaced. To detect the adjacent patterns for a particular shape, the resizing algorithm described in the previous section can be applied with minor modifications.

4.4.4.3. Compensation for pattern generator characteristic error

The pattern generator used to make layout masks has its own characteristic error, resulting in the feature size being different from the original design. By applying the resizing algorithm to the input data for the pattern generator, such errors could be reduced.

4.5. Summary

At the very beginning of this chapter, we describe the geometrical data structure used in HPEX, namely a 4-d binary search tree. The advantage 4-d binary trees have is the efficiency of region and point searchings, both having $O(\log N)$ time complexity. Since these two operations are frequently encountered in layout verification tools such HPEX, employing a 4-d binary search tree in the extraction program should reduce the overall runtime complexity. Following a description of the 4-d binary search tree and its supporting algorithms, we give a detailed implementation of netlist data structure used in HPEX. The main feature of our netlist data structure is that geometric and circuit schematic information are strongly combined together, thereby simplifying extraction of circuit parameters. In addition, since it is easy to access the geometric data from the network data, optimizing the circuit performance by adjusting layout geometries is made feasible.

Next, a new layout resizing algorithm based on scanline approach and a simple rectangle data structure are presented. This resizing algorithm has many applications, such as compensation of process bias, data preparation for E-beam lithography, and reduction of proximity effect. Our resizing algorithm treats layout expansion and contraction differently in order to

improve the efficiency of the algorithm. By using this algorithm for circuit extraction, we can efficiently compensate process variations by contracting or expanding corresponding layout regions. Since circuit parameters strongly depend on layout feature sizes, considering the actual feature sizes after fabrication would substantially improve the accuracy in circuit extraction.

The data structure and algorithms mentioned in this chapter provide a basis for developing geometrical extraction algorithms in HPEX, while resistor and capacitor models derived in Chapter 3 provide a computational method for interconnect *RC* modeling. Beginning in Chapter 5, we will concentrate on circuit extraction and parasitic modeling algorithms used in HPEX and detailed algorithms of hierarchical and flat circuit extractions used in HPEX will also be discussed.

CHAPTER 5.

HIERARCHICAL AND FLAT CIRCUIT EXTRACTION

5.1. Introduction

Hierarchically structured design is a common practice of handling the increasing complexity of VLSI circuits [51]. Layout designers often draw the layout of integrated circuits in the form of a hierarchical specification in which each cell is made up of geometries and references to other cells. The typical approach to circuit extraction involves flattening the hierarchical representation to one level. The program then performs circuit extraction on the flat representation. This eliminates all hierarchical structure present in the layout input, dramatically simplifying the extraction program. However, eliminating the hierarchy increases the extraction time because each instance of the cells has to be extracted. In addition, the memory space required may be very large, since the layout is stored in a flat representation. Therefore, from the extraction time-and-space point of view, it is more efficient to build a circuit extractor which is able to exploit the hierarchy in the circuit layout.

Although hierarchical extraction can save CPU time and memory space, the actual savings strongly depend on the regularity factor in a layout design. This regularity factor is defined as the ratio of the total number of transistors after layout flattening to the number of transistors actually drawn. If this factor is large, the savings can be pronounced. Furthermore, the output format will be more readable if the hierarchy of a circuit layout is taken into account. Our objective here is to develop an efficient hierarchical parasitic circuit extractor.

In the literature, several hierarchical circuit extractors have been reported [52-54, 4, 55-56]. Most of these, however, extract only parasitic capacitances [52-55]. Magic's layout extrac-

[illegible] the procedure for resistance calculation [illegible] the resistances computed by Magic's layout [illegible] Bostenaru et al [56] proposed to use the [illegible] mplementation has been reported.

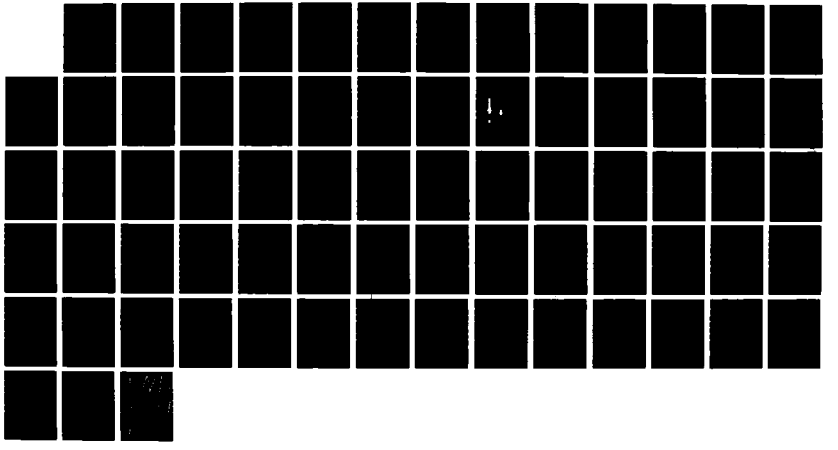
[illegible] extractor HPEX which can model details of [illegible] design methodology embedded in HPEX is dis- [illegible] flexible restricted hierarchy in handling cell overlapping [illegible] extraction algorithms and their implementation [illegible] give some examples to show the perfor-

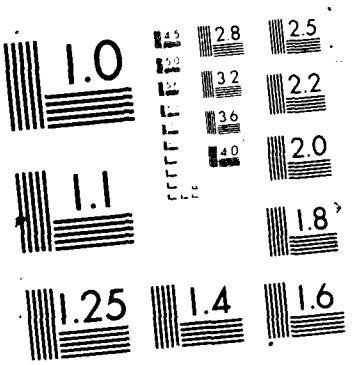
5.2 Design Methodology for HPEX

[illegible] description language (e.g. CIF) cells or symbols are defined as [illegible] boxes and instances of other cells. Following certain cell composi- [illegible] complex cells can then be constructed from lower level cells. It must be noted [illegible] regarded as a cell which is on the top of the hierarchy, i.e., the root [illegible] Since a cell can be instantiated several times in a circuit design, this design [illegible] reducing the total number of geometric primitives needed to be [illegible] Even if all cells are used only once, this kind of language still facili- [illegible] breaking the design into more manageable pieces, thereby reducing the [illegible] The hierarchical structured design is widely adopted today to manage [illegible] VLSI system design including structural and physical designs.

[illegible] described hierarchically, the technique to handle the hierarchy inherent in [illegible] the circuit extraction depends heavily on the layout methodology. It is therefore important to know the layout methodology used by designers, or to choose a particular layout

AD-A185 847 EXTRACTION OF MOS VLSI (VERY-LARGE-SCALE-INTEGRATED) 2/2
CIRCUIT MODELS INCLU..(U) ILLINOIS UNIV AT URBANA COLL
OF EDUCATION S SU SEP 87 UILU-ENG-87-2254
UNCLASSIFIED N00014-84-C-0149 F/G 9/1 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

methodology before developing an extraction program. Several methodologies emphasizing different aspects have been proposed [58]. Generally, design methodologies fall between the following two extremes:

(1) Restricted layout hierarchy :

In each level of the layout hierarchy cells are not allowed to overlap, and geometric primitives also are not allowed to overlap any cell. Under such constraints, cells communicate with the environment only through ports which are defined on the boundary. Another restriction is that no partial devices can be formed on the cell boundary.

(2) Nonrestricted layout hierarchy :

No constraints have been made or assumed in this methodology. Cells or primitives and cells are allowed to overlap whenever necessary. The designer has total freedom in laying out cells and placing them in any preferred position. In general, devices are allowed to be placed right on the cell boundary.

In the restricted design methodology, the cell composition rules are relatively easy because all interactions are around the cell boundary. In addition, the layout hierarchy is exactly the same as the circuit design hierarchy. This makes the circuit extraction simple and straightforward. However, this style of layout looks clumsy and sometimes conservative due to overlapping constraints. In the nonrestricted design methodology, on the other hand, the designer has total control over how to lay out cells and use them. Nevertheless, allowing overlapping between cells often makes the layout hierarchy different from its corresponding circuit hierarchy. This not only degrades the extractor performance, but also makes the consistency check more difficult. In general, there is a trade-off between the adopted design methodology and the CAD performance. Sometimes a little sacrifice of designers' freedom will substantially upgrade the extractor performance, and in turn, shorten the design time and increase productivity.

The layout methodology we choose is closer to the restricted design hierarchy. In order to produce the extracted circuit which has the same layout hierarchy as specified in the input description, we encourage the designers to design circuit layouts with no overlaps between cells, and between cells and primitives. However, our extractor is still able to handle overlapping cells if necessary. A simple method, described as follows, is used to to resolve the cell overlapping problem. After the CIF input file is read in by HPEX, a tree-structure hierarchy as shown in Figure 5.1 is retained in the internal data structure. The minimum bounding box (MBB) of each cell, which is defined as the cell boundary, is also calculated in the input phase. Suppose

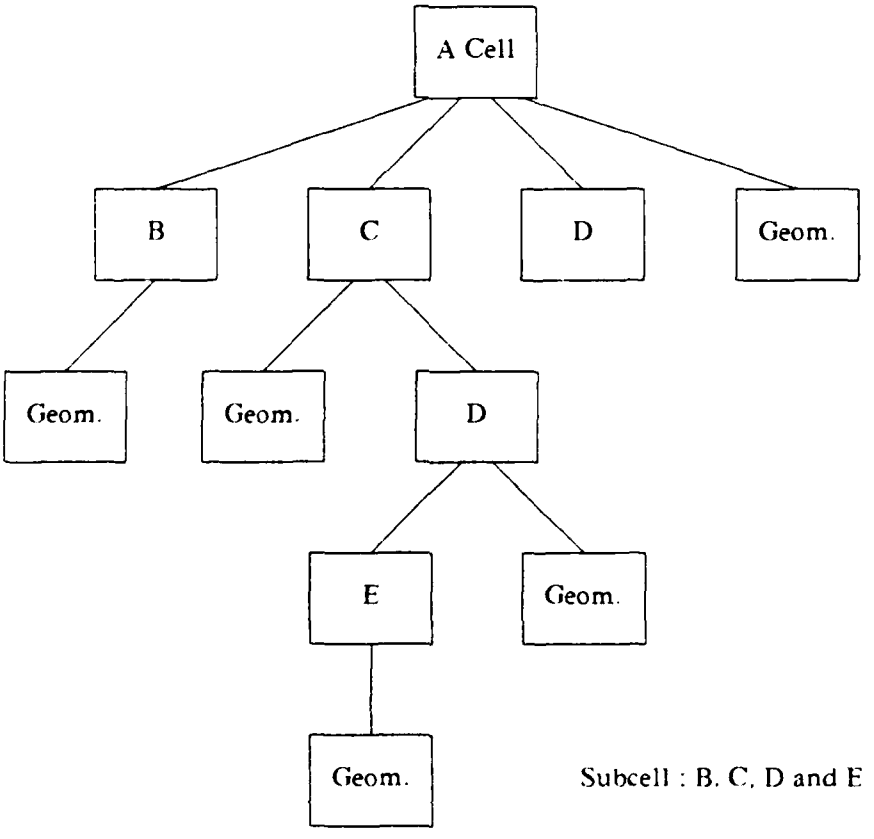


Figure 5.1 A tree hierarchy of an example layout description.

that two overlapping cells were detected; we could flatten one level of hierarchy for each cell and continue this top-down process until no overlapped cell is found. This method can also be applied to the case when one cell is overlapped by the geometrical primitives. The worst case of this method is that the whole design is flattened to one level, and circuit extraction is performed in a nonhierarchical fashion. Thus, in order to increase the performance of our extractor, the designers are encouraged to use as many nonoverlapping cells as possible in the designs.

5.3. Hierarchical Circuit Extraction

5.3.1. Input phase

The front end of HPEX first reads in the layout artwork data defined in the CIF description language, preserving the layout hierarchy. Every cell in the internal data structure contains several linked lists of mask levels and pointers to other cells. Figure 5.2 shows the implementation of the tree hierarchy as shown in Figure 5.1.

The information stored in the cell record also includes the cell boundary and the location and mask level of terminals. Since the cell boundary is not explicitly specified in the input file, the cell boundary we use is defined as the minimum bounding box (MBB) which covers all primitives and subcells of this cell. The MBB is computed during the input phase after all elements are read in. It must be noted that some user-extension commands are defined in the CIF file in order to clearly indicate the cell terminal and net information. This is due to the fact that in the original CIF layout language only the geometric topology is described, and no structural circuit information is provided. For example, power and ground nodes are not known a priori, if we do not provide additional information in the CIF input. This kind of net or node information will be greatly useful and play an important role in the layout verification to be performed later.

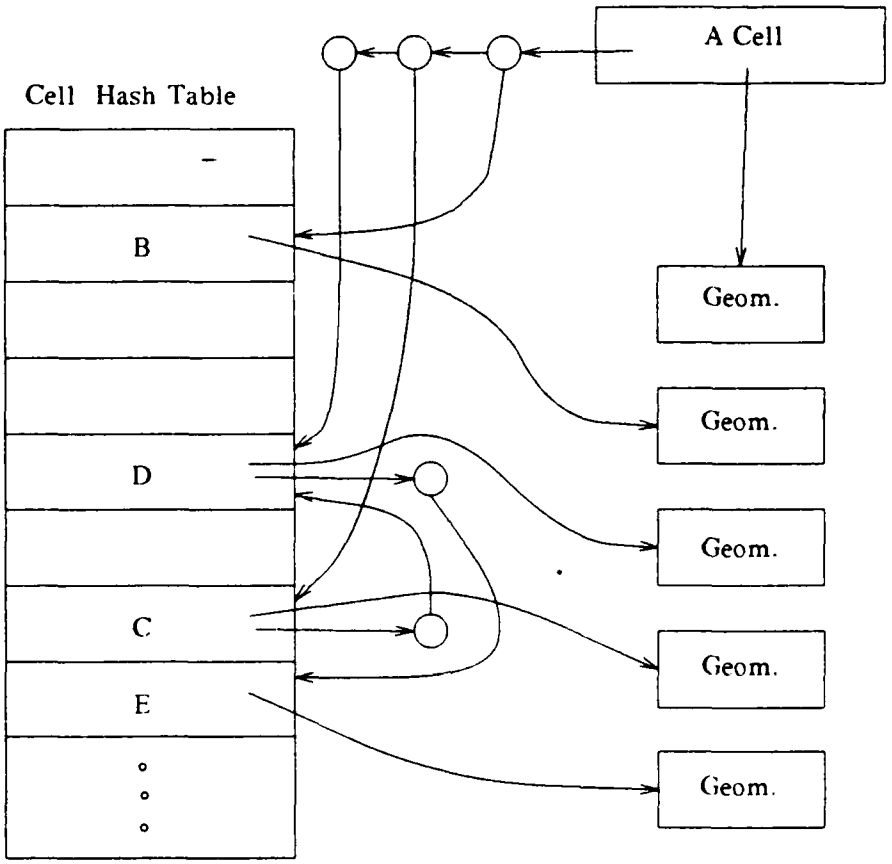


Figure 5.2 Internal data structure of HPEX corresponding to that shown in Figure 5.1.

5.3.2. Analysis of layout hierarchy

In order to adapt the input layout to our design methodology, the layout hierarchy should be analyzed and reconstructed. The main purpose in reconstructing the hierarchy is to remove all possible overlappings between instantiated cells and cell and geometric primitives. The hierarchically recursive tree walk algorithms employed in HPEX for restructuring the layout hierarchy are described as follows:

```
RebuildTree(A_Cell)
begin
   1. List := A_Cell^.subcell;
     while (List <> nil) do
     begin
       Subcell := ThisCell(List);
       if (Subcell is not checked before) then
          begin
             RebuildTree(Subcell);
             Set Subcell to "checked" status;
          end;
       List := NextElement(List)
          end;
     end;
   2. RemoveCellOverlap(A_Cell);
end;

RemoveCellOverlap(A_Cell)
begin
   1. List := A_Cell^.subcell;
     while (List <> nil) do
     begin
       Subcell := ThisCell(List);
          if (Subcell overlaps other subcells or
                geometric primitives) then
          begin
                   Put Subcell in a list ExpandList;
                   Delete Subcell from List;
          end;
       List := NextElement(List)
     end;
   2. List := ExpandList;
     while (List <> nil) do
       begin
       Subcell := ThisCell(List);
          Expand Subcell in A_Cell by one level of hierarchy;
          List := NextElement(List);
       end;
   3. if (ExpandList <> nil) then
          RemoveCellOverlap(A_Cell);
end;
```

All steps described in the above algorithms are self-explanatory. The procedure "RebuildTree" hierarchically removes all overlapping subcells inside a given cell by a bottom-up fashion, while the procedure "RemoveCellOverlap" detects and expands overlapping cells recursively. Once we apply the procedure "RebuildTree" to the original tree hierarchy, the output tree hierarchy will then be ready for the next step processing.

5.3.3. Hierarchical extraction

The layout data can be sent to the hierarchical extraction module after the input layout is read in and preprocessed by HPEX. The hierarchical circuit extraction algorithm used in HPEX can be described in the following PASCAL-like pseudocode:

```
Circu...act(A_Cell)
begin
   1 List := A_Cell^.subcell;
     while (List <> nil) do
     begin
       Subcell := ThisCell(List);
       if (Subcell is not extracted before) then
             CircuitExtract(Subcell);
         Set Subcell to the "extracted" status;
       List := NextElement(List)
     end;
   2. Register the location and mask level of segments
     for corresponding terminals of A_Cell;
   3. List := A_Cell^.subcell;
     while (List <> nil) do
     begin
       Subcell := ThisCell(List);
       Instantiate all terminal segments of Subcell in A_Cell;
       List := NextElement(List)
     end;
   4. FlatExtract(A_Cell)
end;
```

Since the bottom-up cell processing is employed in the hierarchical circuit extraction, the cell cannot be processed unless all its subcells are already extracted. Therefore, for a particular cell design the first step in the algorithm is to check if all its subcells are already extracted. If there are some subcells which are not extracted before, recursive calls for circuit extraction will be invoked for these unextracted subcells. Otherwise, this cell goes on to the next step for further processing.

After step 1 in the above algorithm, the cell under processing is ready for circuit extraction. However, some preprocessing steps must first be taken in order to take the boundary information of subcells into consideration. These steps are primarily for the registration of information around the cell boundary. The cell boundary information is propagated from the

lower level hierarchy to its parent cell during circuit extraction, including the segments for the subcell terminals instantiated in the parent cell and the sequence for these segments. The rectangle data structure employed in our extractor easily represents any segment by setting either the length or the width to zero. The sequence used for the instantiated segments is important and will be used to register the external node numbers for the subcircuit calls when the circuit extraction is finished.

The registration of cell boundary information is mainly done in steps 2 and 3. Step 2 is to find out and register the terminal information of the cell currently under extraction for later references. As for step 3, all terminal information about the subcells is instantiated and registered in the parent cell in order to correctly interpret the information contained in the parent cell. In this instantiation the space transformation is necessary and can be calculated according to translation, rotation and mirroring as specified in the CIF input. Since we only deal with Manhattan type layouts, rotation of cells other than multiples of 90 degree is prohibited, and mirroring of cells is allowed only with respect to the $x-$ or $y-$axis. It should be noted that in step 3 new geometries corresponding to all terminal segments of subcells are also created and added to the cell under processing for the terminal registration. Once these preprocessing steps are finished, the cell is then sent to the flat extractor module for the transistor and parasitic extraction.

5.4. Flat Circuit Extraction

Flat circuit extraction in HPEX can be illustrated by the flowchart shown in Figure 5.3. The artwork data is first preprocessed according to process variations by layout resizing algorithms described in Chapter 3 . By applying boolean mask operations, MOS transistors and interconnection geometries are efficiently and correctly identified. This step is called geometrical extraction, generally consisting of two major steps: (1) finding active devices from certain combinations of mask layers specified in the technology, and (2) identifying electrically

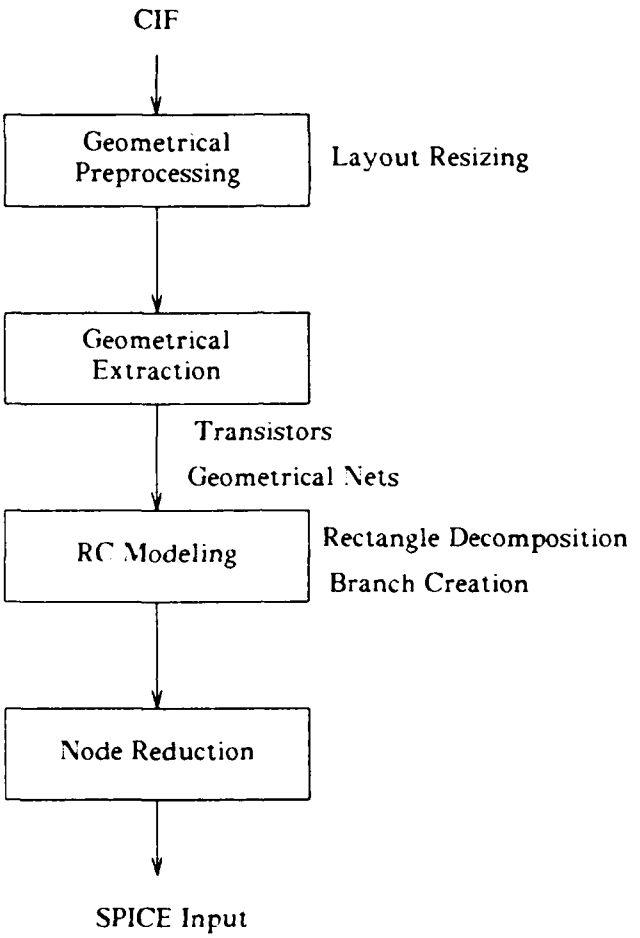


Figure 5.3 Flowchart for HPEX to generate SPICE file.

connected nets. Following this, detailed interconnect parasitics are extracted by feeding geometrical nets into the *RC* modeling module. Finally, an accurate node reduction algorithm is applied to reduce the number of parasitic elements. In this chapter, we will mainly focus on general flat circuit extraction procedures and the *RC* modeling. The detailed node reduction method will be discussed in the next chapter.

5.4.1. Extraction procedures

In HPEX, flat circuit extraction procedures can be summarized as follows:

Circuit extraction steps

E1. Read all geometric primitives specified in the cell.

E2. Resize the input layout data by user's specified process bias and construct 4-d binary search trees [45] for each mask level;

E3. Find all overlaps between diffusion and poly rectangles to determine transistor channels by excluding butting and buried contacts, then remove all channels from diffusion rectangles;

E4 Use depth-first search to group channels which belong to one transistor and from transistor channels find rectangles associated with the drain or the source;

E5. Use information about transistor source, drain and gate to find all nets by depth-first search through all electrically connected rectangles;

E6. Find all power and ground nets from user-specified coordinates in CIF input file;

E7. Feed all nets except power and ground nets into interconnect *RC* model module to compute associated resistances and self-capacitances;

E8. Compare nets pairwise to compute all possible coupling capacitances;

E9. Find all i/o, power, or ground node numbers from user-specified coordinates;

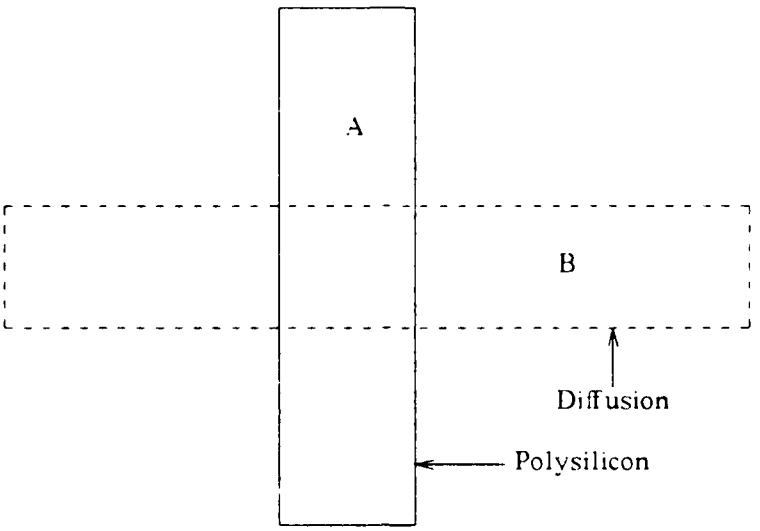
E10. Perform the node reduction on some of the nets;

E11. Report a network composed of circuit elements, such as transistors, resistors and capacitors, and its i/o, power and ground node numbers.

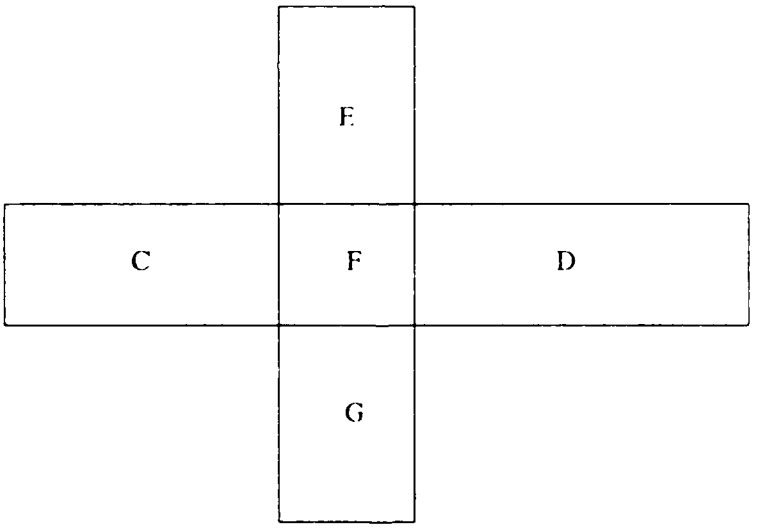
In the extraction step E2, the 4-d binary trees for diffusion and poly regions have to be updated because channel regions have to be removed from diffusion and poly trees. In this way, not only transistors with simple rectangle gates but also those with circular or serpentine gates can be identified. An example illustrating how to update diffusion and poly trees is shown in Figure 5.4 where new diffusion rectangles C and D and poly rectangles E, F and G need to be inserted in the trees while two original rectangles A and B are marked inactive. If there is a long poly rectangle intersecting several diffusion rectangles, the decomposition of this poly rectangle into smaller rectangles is required in order to separate channels from interconnect polys. In the decomposition algorithm, the quick sort algorithm is first applied to find an increasing order of channel segments, and then by this order we can easily decompose the original rectangle.

5.4.2. RC modeling module

In interconnect *RC* modeling, the first step is to search for all nets in which each net contains many rectangles. Then from the net information an *RC* network model will be generated in step E6. If all resistances are neglected in one net, all rectangles in this net should be equipotential. But for detailed modeling of the electrical behavior of the interconnects, resistances must be included. The difficulty associated with the resistance extraction is that resistances are strongly dependent on the current flow in one net. However, the information about current flows is unknown before actually simulating the circuit. In order to achieve greater accuracy we therefore have to preprocess mask data information in the net since only simple resistance formulas are used in our extractor. In preprocessing mask data, the rectangles which is electrically abutted by other rectangles are decomposed into a series of rectangles. The rectangle decomposition algorithm we use is described as follows:



Mark rectangles A and B inactive.



Insert new rectangles E, F and G into the poly tree,
and new rectangles C and D into the diffusion tree.

Figure 5.4 Update diffusion and poly trees.

Rectangle Decomposition Algorithm

Input : a rectangle A in one net.
Output : decomposition of the rectangle A according to its environment.

begin
R1. Suppose that segment $\overline{A_1A_2}$ is the wider side
of rectangle A. Find all butting segments $\overline{B_1B_2}$, $\overline{B_3B_4}$, ..., and $\overline{B_{n-1}B_n}$
according abutting rectangles of the rectangle A.
R2. Merge and Sort segments $\overline{B_1B_2}$, $\overline{B_3B_4}$, ...
and $\overline{B_{n-1}B_n}$ into an increasing order of segments
$\overline{C_1C_2}$, $\overline{C_3C_4}$, ..., and $\overline{C_{k-1}C_k}$.
R3. From segments $\overline{C_1C_2}$, $\overline{C_3C_4}$, ...
$\overline{C_{k-1}C_k}$ and $\overline{A_1A_2}$ decompose the rectangle A.
end;

As an example, Figure 5.5 shows how the rectangle can be decomposed into several rectangles by the above algorithm. Notice that the quick sort algorithm is used in step R2.

Before turning to the algorithm for branch creation, we need some definitions. If two rectangles have more than one electrically connected point, they are defined to be *electrically connected*. If the rectangle electrically connects more than two rectangles, it is said to be a *knot*. Also, the rectangles associated with the drain, source and gate of any transistor are defined as *ports*, which need not be decomposed. For convenience, we define the rectangle which has only one electrically connected rectangle as a *knot*, since a node number should be created for this rectangle in modeling interconnect parasitics. After knots and ports are defined in one net, they are assigned different integer numbers. The branch creation algorithm, described below, is then applied to find all branches in the net.

Branch Creation Algorithm

Input : a net N.
Output : all branches in this net N

begin
B1. Create an electrically connected graph based on rectangles in the net N;
in this graph, every node represents a rectangle and edges
indicate interconnect information between rectangles.
B2. From any port (or knot) with unmarked edges, find every path ending
at the other port (or knot); and define this path as a *branch*.
Then mark all traversed edges.
B3. Find another port (or knot) with unmarked edges, then go to step B2;
otherwise, return.
end;

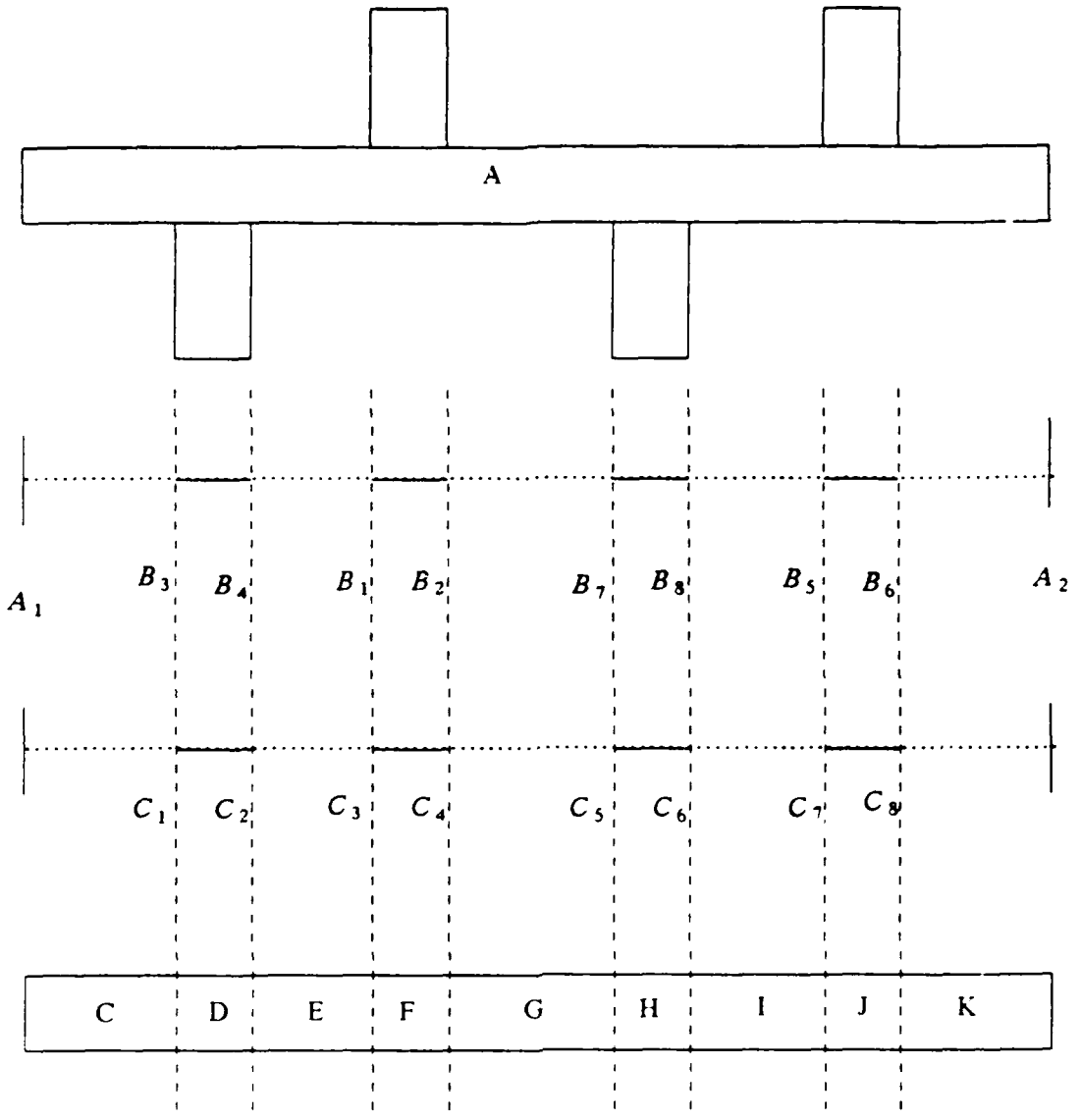
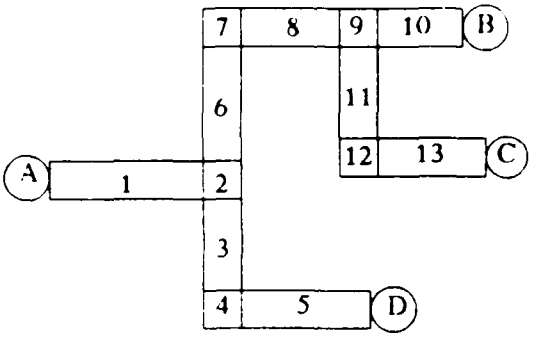


Figure 5.5 Rectangle decomposition.

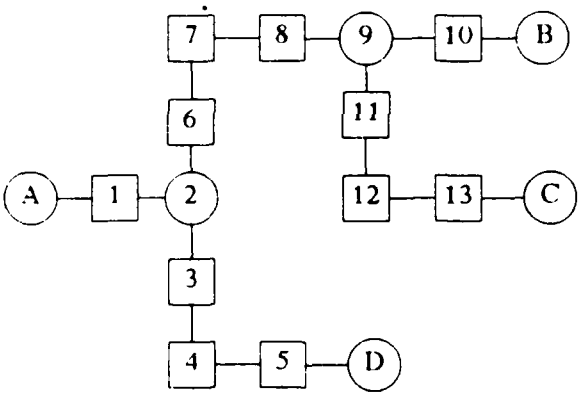
In Figure 5.6, we have an example to show how to use the above algorithm to create all branches in a net. Every branch consists of a list of rectangles in one path, and this information is extremely useful in modeling interconnect parasitics. It should be mentioned here that the information about all branches in one net is stored in the *branch* pointer in the net record.



2 and 9 : knot

A, B, C , and D : port or knot

Electrically Connected Graph



Created Branches

(1) A 1 2

(2) 2 3 4 5 D

(3) 2 6 7 8 9

(4) 9 10 B

(5) 9 11 12 13 C

Figure 5.6 Branch creation.

The next step is to perform resistance and capacitance calculations. However, corner rectangles have to be first detected in order to accurately model resistance of bends. These corner rectangles could be easily found as long as two adjacent rectangles are known. In order not to

overcount resistance, resistances of the rectangles associated with ports and knots are also carefully approximated. For knots and gate ports only half of the resistance is taken into account in resistance calculation, while for diffusion ports the resistance is not computed due to the fact that a circuit simulator such as SPICE2 usually will generate this kind of information internally. Once the resistances of all rectangles in one branch are known, their summation is clearly the resistance of the branch. For self-capacitance calculation, empirical formulas described in the preceding chapter are used; however, equivalent dielectric constants are taken into formulas if there are layers of different dielectrics. Note that self-capacitance of one branch is calculated when computing the branch resistance. Finally, π-lumped circuit model is used to approximate distributed RC behavior of all branches. Another feature of this step is that threshold values R_{th} and C_{selfth} are specified to filter out trivial resistances and capacitances. Therefore, all nodes with zero resistance in between have to be collapsed into one node; the depth-first search algorithm is used to find these nodes.

Before calculating the coupling capacitance between two nets, their minimum bounding boxes are compared to decide whether the coupling is too small to calculate. Since branches are basic elements in calculating the coupling capacitances, their minimum bounding boxes are also compared in order to filter out small couplings. Once the coupling capacitance between two branches is determined, the coupling capacitor is inserted between them through the T-like lumped circuit model. The reason to use this model is that it is easier to insert the coupling capacitance, and at most one more node needs to be created in each branch. Note that since the self-capacitance calculation is separated from the coupling capacitance calculation in our approach, some errors will be produced because both capacitances are closely related. A general increase in the coupling capacitance decreases the self-capacitance. However, the effect of this trend is considered as second order and should not affect the circuit performance significantly.

5.5. Examples and Discussion

All algorithms and techniques described in the previous sections have been implemented in HPEX. The program is coded in PASCAL, which runs on a SUN 3/75 workstation using the UNIX operating system and has about 11000 lines of codes, including program statements and documentation.

5.5.1. Example 1: Extraction and simulation of NMOS one-bit full adder

A schematic circuit diagram of a full adder is shown in Figure 5.7. This circuit was laid out in 3μm NMOS technology. Before extraction, several process dependent parameters must be

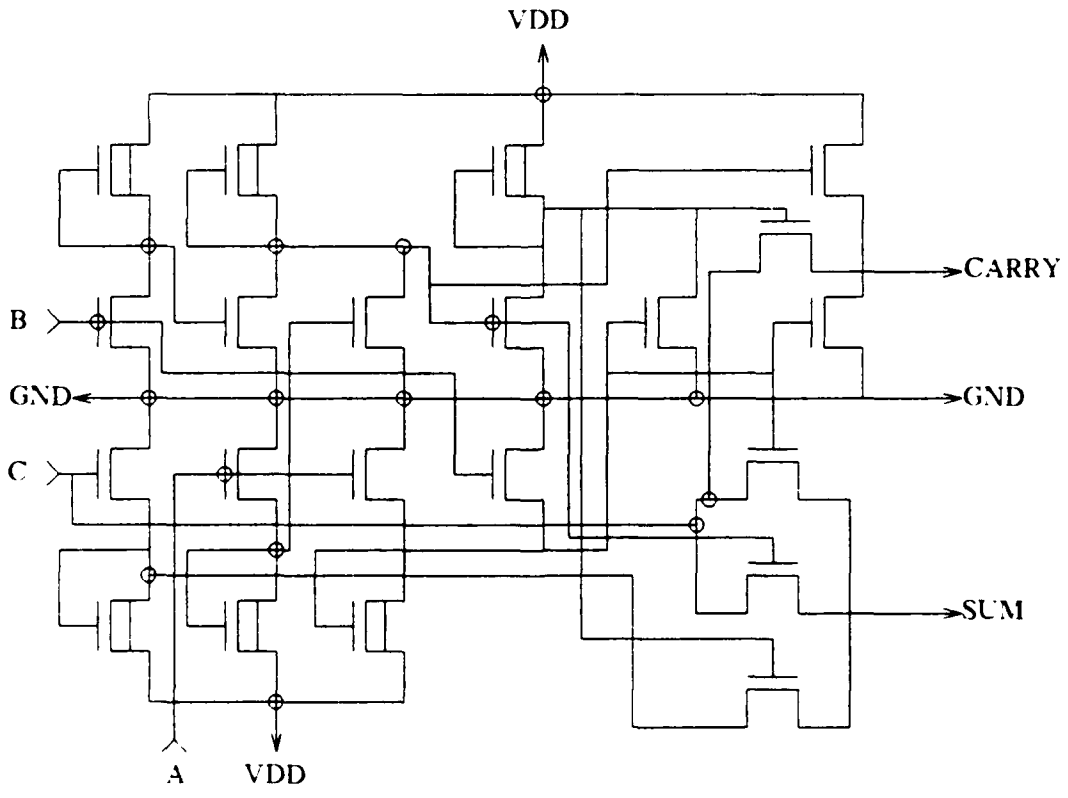


Figure 5.7 An NMOS one-bit full adder.

supplied, such as the sheet resistance and thickness of different conductors and dielectric constants. Shown in Figure 5.8 is the simulation output of the extracted circuit corresponding to

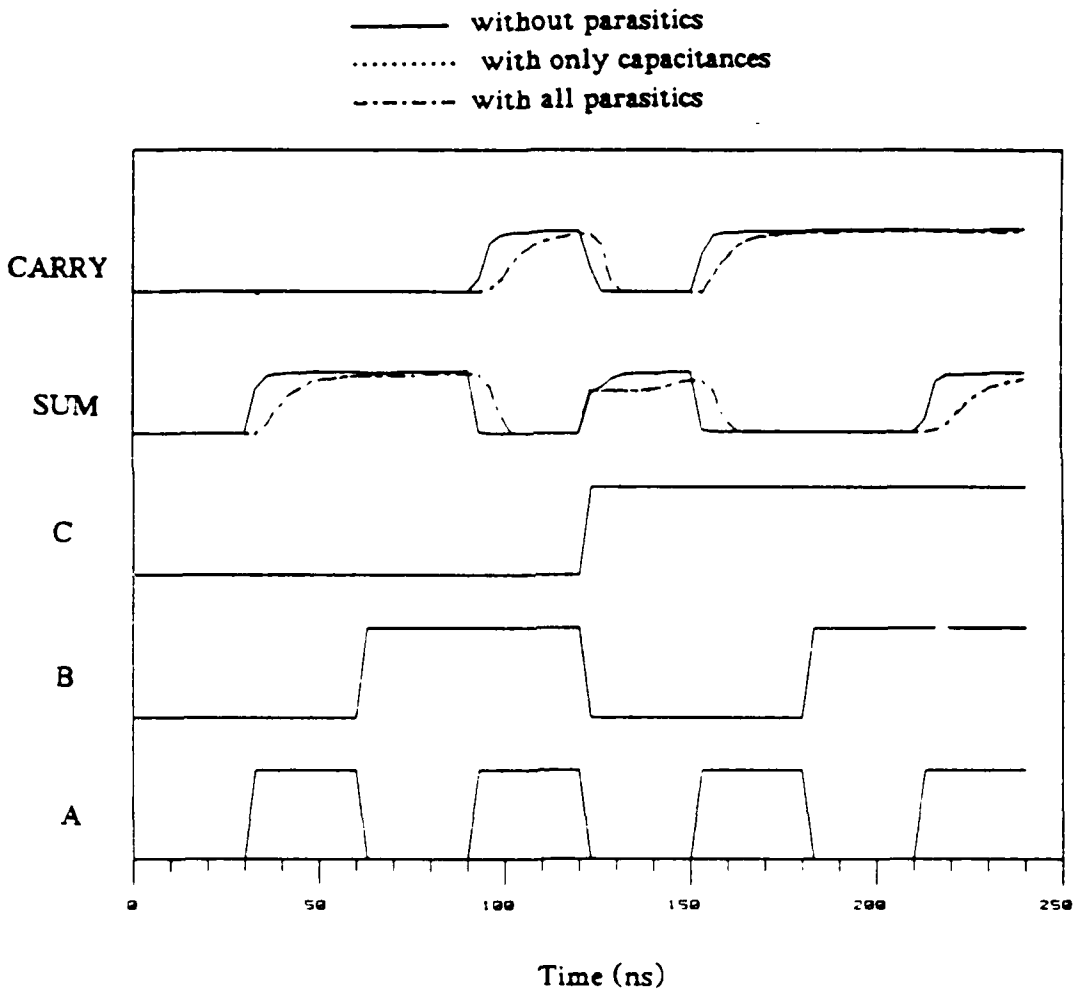


Figure 5.8 Simulation output waveforms for one-bit full adder: (a) without interconnect parasitics; (b) with all interconnect parasitics; (c) with only capacitances.

input waveforms A, B, and C. There are three output waveforms for *SUM* and *CARRY*. The solid lines represent the waveforms without interconnect parasitics, while the dotted lines are the simulation output of the extracted circuit including all interconnect parasitics. In addition, the circuit with only interconnect capacitance is simulated in order to estimate the effect of interconnect resistance. These output waveforms are almost identical to those with all interconnect parasitics. Basically, the three output waveforms are very similar in shape, except for signal delays which are different. The longest delay is observed in the circuit with all interconnect parasitics. If the coupling capacitances are large enough, sometimes glitches or spikes might appear in the output waveform. But this is not the case in this example. Thus it can be concluded that the coupling capacitances are too small to seriously affect the performance of this circuit. Furthermore, the interconnect resistances also have no major effect on circuit performance and therefore can be neglected in this example.

5.5.2. Example 2: CMOS PLA

The PLA layout of this example is generated by PANDA [37], realizing the following function.

$$\bar{s} = z(\bar{x}\bar{y} + xy) + (\bar{x}y + x\bar{y})\bar{z}$$

$$\bar{c} = xy + y\bar{z} + x\bar{z}$$

where x, y and z are inputs, and s and c are outputs. Although PLA circuits produced by PANDA are implemented in the CMOS technology, they employ NMOS circuit design concept, i.e. most PMOS transistors are used as pull-up resistors and their gates are tied to ground. The extracted circuit of this example with all interconnect parasitics consist of 15 p-type transistors, 29 n-type transistors, 114 resistors and 150 capacitors. For comparison, two other circuits with different details of interconnect parasitic are also extracted. Simulation results showed that, in this example, the effect of interconnect parasitics is still too small to seriously affect the circuit performance.

5.5.3. Example 3: NMOS serial adder

The layout of this serial adder is obtained from [38]. It is realized in 3μm NMOS technology and its logic diagram is shown in Figure 5.9. The extracted output contains 41 transistors, 90 resistors and 117 capacitors. Simulation results show that interconnect resistances are too

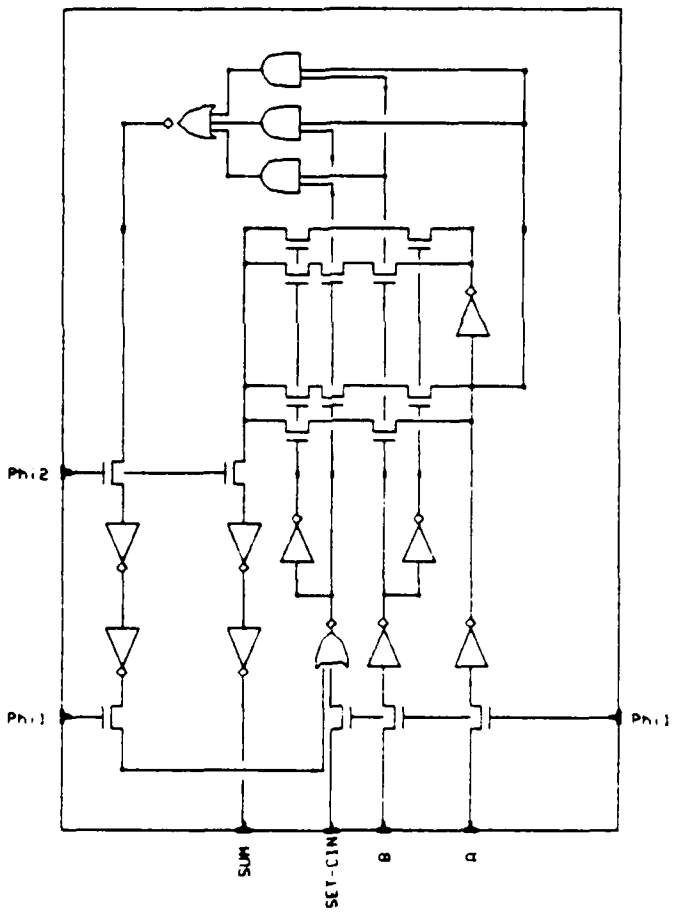


Figure 5.9 An NMOS serial adder.

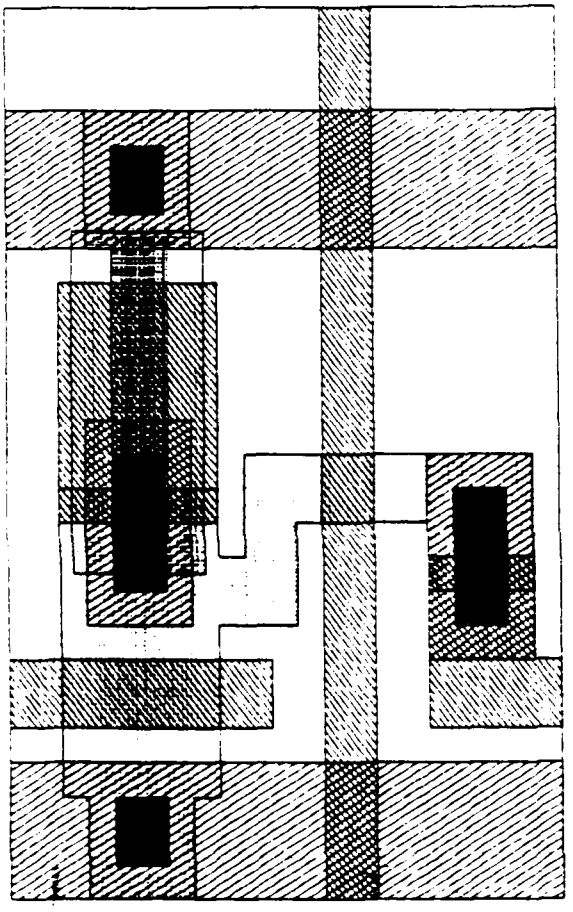
small to degrade the timing performance, while interconnect capacitances indicate a significant impact on the signal delay. For one particular input vector, the critical path delay increases from 18.6 *ns* to 22.55 *ns* by adding all parasitic capacitances.

From the above three examples, we conclude that the effect of interconnect parasitics on circuit performance is not only layout-dependent, but also technology-dependent. Signal delays in circuits are usually affected by self-capacitances and resistances. However, coupling capacitances might produce glitches or spikes and their magnitude is heavily dependent on the actual circuit layout. In addition, the effect of capacitive coupling noise is not known until the extracted output is simulated. Therefore, in circuit designs one must have the ability to estimate the coupling capacitances between signal lines, and take their effect into account.

5.5.4. Example 4: Hierarchical extraction of NMOS half-shift register

An example circuit layout for a half-shift register and its corresponding extracted output appear in Figure 5.10. Since the output has the same format as the SPICE input, we can easily perform circuit simulations on this output. A point to be noted here is that the process file which contains process information such as oxide and conductor thicknesses is separated from the executable file. As a result, different process parameters for any given layout can be easily substituted to predict performance variation due to different process lines.

Runtimes of a set of idealized layouts which are composed from the dynamic shift register shown in Figure 5.10 are listed in Table 5.1. Each successive circuit layout consists of four instances of the previous layout with no overlapping cells. The runtime for extraction with coupling capacitances is proportional to the perimeter of the cells plus the factor due to checking steps in handling the layout hierarchy and calculating coupling capacitances. However, the runtime for extraction without coupling capacitances is proportional to the perimeter of the cells and the factor only due to preprocessing steps in handling the layout hierarchy. It is



```
*half shift register
.subckt shre1  19 19 27 27 32 36 40 41
*gndr : 19
*gndl : 19
*vddr : 27
*vddl : 27
*out : 32
*in : 36
*phiu : 40
*phib : 41
r0001     39    42   9.000e+01
r0002     42    41   9.000e+01
r0003     39    43   1.050e+02
r0004     43    40   1.050e+02
c0001     40     0   0.00341pf
c0002     41     0   0.00288pf
c0003     39     0   0.00629pf
c0004     36     0   0.00157pf
r0005     28    32   1.353e+01
c0005     28     0   0.00996pf
c0006     32     0   0.00187pf
r0006     24    27   1.255e+01
c0007     27     0   0.02089pf
c0008     19     0   0.01928pf
c0009     10     0   0.02818pf
c0010     27    43   0.00183pf
c0011     19    42   0.00183pf
m1     10    10    24 0 dpl  l= 12.00u  w= 4.00u
+ ad= 24.00p  as=  8.00p
m3     28    39    10 0 enh  l= 4.00u  w= 4.00u
+ ad= 48.00p  as= 24.00p
m2     19    36    10 0 enh  l= 4.00u  w= 12.00u
+ ad= 48.00p  as= 24.00p
.ends  shre1
```

Figure 5.10 Layout of an NMOS half-shift register and HPEX output.

Table 5.1 Runtimes for different compositions of a half-shift register.

Layout	Flat transistors	Runtime (sec) (with coupling capactances)	Runtime (sec) (without coupling capacitances)
HalfShift	3	2.37	2.12
2x2	12	2.82	2.43
4x4	48	3.97	3.38
8x8	192	6.63	5.33
16x16	768	14.22	8.65
32x32	3072	37.15	16.92
64x64	12288	113.18	35.08
128x128	49152	387.85	75.15
256x256	196608	1427.28	172.90

observed that the runtime for extraction with coupling capacitances becomes pronounced compared to that for extraction without coupling capacitances when a layout has a large number of transistors and a deep hierarchy. This phenomenon can be attributed to the fact that time complexity $O(N^2)$, where N is the number of nets, for computing coupling capacitances plays a deciding role in the total time complexity when the layout is significantly large.

5.5.5. Example 5 : Flat extraction of NMOS one-bit full adder

Table 5.2 shows the runtimes of flat extractions for different numbers of one-bit full adder circuits as illustrated in Example 1. Note that the number of capacitors as shown in the table contains both self-capacitances and coupling capacitances. Experimental data indicate that time complexity $O(N\log(N))$, where N is the number of boxes, of 4-d binary search trees [45] used in geometrical extraction will dominate extractor performance. Therefore, in order to improve extractor performance, the scanline approach for geometrical circuit extraction is recommended since its observed time complexity, in general, is $O(N)$ where N is the number of boxes in the layout. Another observation is that for moderate sizes of cells the runtime for

Table 5.2 Runtimes for different numbers of one-bit full adder.

Trans.	Capac.	Resis.	Preproc. time (sec)	Geom. extr. time (sec)	Interconn. modeling time (sec)	Total time (sec)
21	104	66	1.92	3.57	9.88	15.37
42	212	134	2.45	9.23	19.57	30.45
63	318	202	3.13	16.82	28.92	48.87
84	424	270	3.87	26.25	38.82	68.93
105	530	338	4.67	37.68	48.40	90.75
126	636	406	6.18	56.78	64.90	127.87

interconnect modeling is almost linearly proportional to the number of boxes specified in the circuit layout, although some algorithms in modeling interconnects require $O(N^2)$ time complexity, where N is the number of nets in the layout. From this observation, we conclude that the ideal input for our extractor would be a layout composed of moderate sizes of cells, each of them containing a few hundred transistors.

5.6. Comparison with Magic's Layout Extractor

Magic is an interactive layout system developed by J. K. Ousterhout et al. [59] for MOS custom integrated circuits. The system incorporates a set of design and verification tools: a layout editor, a continuous design rule checker, a plowing tool, a router, and a circuit extractor. Since circuit extraction is our main concern, we only discuss the comparison between HPEX and Magic's extractor in this section.

In order to compare HPEX with Magic's extractor, some circuit layouts have been edited and extracted in the Magic layout system. These layouts are then extracted by HPEX. From our experience with HPEX and Magic's extractor, several differences between these two extractors are observed:

(1) Computer Language - HPEX is written in PASCAL, while Magic's extractor is coded in C.

(2) Data Structure - HPEX uses a 4-d binary search tree for the internal data structure, while Magic's extractor is based on a data structure called *corner stitching* [60]. From the CPU time point of view, it indicates that Magic's extractor runs faster than HPEX. This is because *corner stitching* is more efficient for geometrical searching operations in the connectivity extraction. Furthermore, the code in HPEX is not optimized. However, Magic's extractor needs more memory space due to the nature of the *corner stitching* data structure; the entire layout space including both layer and empty spaces is explicitly represented.

(3) Circuit Model - HPEX describes the connections between transistors as a detailed *RC* network. The extracted transistor netlist along with detailed *RC* networks are used for timing analysis or detailed circuit simulation. A simpler circuit model is adopted in Magic's extractor. It models transistor interconnections as nodes. A node is like an equipotential, but it includes a lumped parasitic resistance and self capacitance. Although this node model is fairly simple and efficient, it neglects the distributed behavior of interconnection lines by only computing a lumped resistance. In addition, a very simple approximation is used in Magic's extractor to calculate lumped resistances. This approximation can be described as follows. The total perimeter and total area of each type of layer comprising a node are first computed. Then the node is assumed to be a simple rectangular region. The resistance is simply obtained by solving a quadratic equation from the perimeter and area. Although this approximation is straightforward in computation, it performs poorly for a node with many branches, for example, an interconnect layout and its corresponding extracted resistance networks as shown Figure 5.11. It clearly indicates that the HPEX output network is more accurate compared to the resistance network generated by Magic.

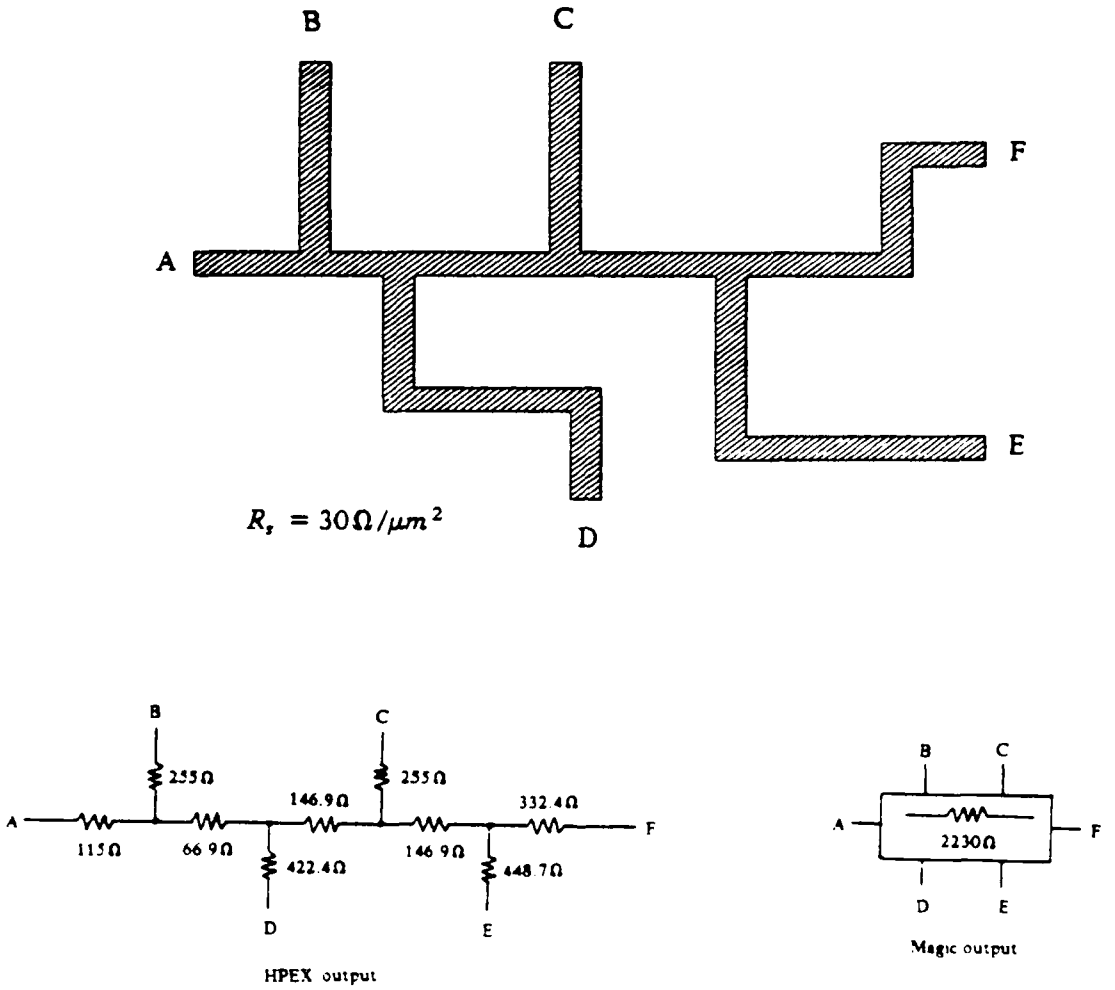


Figure 5.11 An interconnect layout and extracted HPEX and Magic outputs.

(4) Hierarchical Extraction - HPEX hierarchically extracts a circuit layout based on non-overlapping cells. All overlapped cells have to be flattened in this type of extraction. Magic's extractor, on the other hand, allows overlapped cells as long as no transistors are

created or destroyed. The reason for more flexible extraction in Magic is that the parasitic extraction of Magic's extractor based on a node model is easier to implement as compared to that of HPEX.

(5) Output Format - HPEX produces the output which is exactly SPICE-compatible. However, Magic's extractor generates its own output format. If a timing analysis program such as CRYSTAL is used to locate a critical path, the Magic extracted format should be translated into the CRYSTAL input by a program called *ext2sim*. In this translation, all extracted cells are expanded into a file consisting of transistors, resistors and capacitors. No hierarchical information is retained in the output by such translation, and this is contrary to the HPEX output. There is also a program called *sim2spice* which transforms the CRYSTAL input into the SPICE input. Unfortunately, this program does not recognize the resistance elements. Therefore, in the SPICE input file all resistances are discarded. This further degrades the accuracy of the interconnect model.

In summary, HPEX is better in terms of the parasitic circuit model, but Magic's extractor performs well in terms of speed. One suggestion for future improvement may be the implementation of HPEX parasitic models in Magic or the implementation of the *corner stitching* data structure in HPEX.

CHAPTER 6.

NODE REDUCTION TECHNIQUE

6.1. Introduction

Algorithms to extract lumped *RC* circuits from the layout of interconnect regions have been developed in Chapter 5. The number of extracted parasitic elements strongly depends on the actual physical layout and the lumped *RC* circuit models used in circuit extraction. However, in general, a large number of parasitic elements as compared to the number of active devices will be extracted after circuit extraction. In order to predict the variation of the timing performance caused by introducing parasitic elements, the transistor netlist along with the parasitic lumped *RC* networks have to be simulated by circuit simulators. If we include all these elements in the circuit simulation file, the simulation time will be increased. Furthermore, from the verification standpoint, it is very difficult to pinpoint and isolate the effect of parasitic elements when we simulate the circuit file including all interconnect parasitics. For example, if after circuit simulation we find that there is a timing performance degradation due to some parasitic elements, detecting these elements would be sometimes impossible because of a large number of parasitic elements included in the simulation file.. Therefore, by taking the simulation time and verification effort into account, in parasitic extraction we need a node-reduction technique which is able to accurately reduce complicated parasitic *RC* networks into simple lumped circuit networks.

A brute-force approach for the node (or element) reduction in the extracted circuit output has been proposed by Bastian et al. [5]. In their approach only rule-of-thumb heuristics are applied to reduce the number of parasitic elements. For instance, capacitance and resistance threshold values are set to filter out some elements with smaller values. The positions of

resistors or capacitors are flipped in order to combine two resistors or capacitors. Although this type of approach is simple and acceptable in the sense of locality, it often produces somewhat inaccurate results if we take the global *RC* network into account. One such example is that the summation of a large number of small resistances may have a strong effect on the signal delay, regardless of the small values of individual resistances. Instead of using pure heuristics we must consider the signal delay of *RC* networks in developing an accurate node-reduction technique. Signal delay in *RC* trees or general *RC* networks has recently attracted much attention and interest [61-62] because of its application in timing analysis of digital MOS integrated circuits, where MOS transistors are approximated by linear resistors. Generally, the signal delay through *RC* circuits cannot be calculated in closed form. Without using a circuit simulation technique, two approaches are proposed to analytically estimate the signal delay through *RC* networks. The first one [61] finds closed-form delay bounds at output node in *RC* networks, while the second one [62] roughly uses a time constant to approximate the signal delay through *RC* networks. The advantage of the first method is that the error bound can be predicted if delay bounds are applied to estimate the timing information. However, one time-constant approach, though hard to predict the error bound, is simple in the delay calculation.

In this chapter, a new accurate node-reduction technique based on a single time-constant approach to reduce *RC* tree networks is presented. By applying this technique, not only the number of parasitic elements can be reduced, but also the delay effect caused by parasitics can be accurately retained. This chapter is organized as follows. In order to describe the node-reduction problem, some definitions associated with a linear *RC* network are first given. Following this the node-reduction method for a timing estimation is described. Based on the results derived for timing estimation, we then develop a heuristic node-reduction algorithm for the purpose of circuit simulation.

6.2. Definitions

Definition 6.1 RC tree

A tree network of resistances on N+1 nodes, not including the ground node, with node 0 designated as the *root* and C_k denoting the capacitance connected between node k and ground, is called an *RC* tree.

Definition 6.2 R_{ij} in an RC tree

Let P_i be the path between the root and node i in the *RC* tree. Define P_{ij} to be $P_i \bigcap P_j$. Then R_{ij} is defined as the sum of resistances in path P_{ij}. If there is no intersection between paths P_i and P_j, then R_{ij} is zero.

Definition 6.3 Elmore's time constant [63]

In a linear *RC* network with zero initial charge, *Elmore's time constant* of node k is defined as

$$T_{Dk} = \int_0^\infty t y'_k(t)dt \ , \tag{6.1}$$

where $y'_k(t)$ is the derivative of the transient response $y_k(t)$ of node k in the network. For an *RC* tree, Elmore's time constant of node k can be derived as

$$T_{Dk} \overset{\Delta}{=} \sum_{i=1}^{N} R_{ki} C_i \ . \tag{6.2}$$

Definition 6.4 P-R bounds of an RC tree [61]

Consider a unit step input applied to the root node in an *RC* tree at time $t=0$. The following lower and upper bounds $v_{lk}(t)$ and $v_{uk}(t)$ of the voltage waveform $v_k(t)$ at output node k are defined as *P−R bounds*.

$$v_{lk}(t) = \begin{cases} 0 & t \leq T_{Dk} - T_{Rk} \\ 1 - \dfrac{T_{Dk}}{t + T_{Rk}} & T_{Dk} - T_{Rk} \leq t \leq T_P - T_{Rk} \\ 1 - \dfrac{T_{Dk}}{T_P} \exp\left\{(T_P - T_{Rk} - t)/T_{Dk}\right\} & T_P - T_{Rk} \leq t \end{cases} \tag{6.3}$$

$$v_{uk}(t) = \begin{cases} 1 - \dfrac{T_{Dk} - t}{T_P} & t \leq T_{Dk} - T_{Rk} \\ 1 - \dfrac{T_{Rk}}{T_P} \exp\left\{(T_{Dk} - T_{Rk} - t)/T_{Rk}\right\} & T_{Di} - T_{Rk} \leq t \end{cases} \tag{6.4}$$

where T_{Dk} is Elmore's time constant, and T_P and T_{Rk} are defined as follows:

$$T_P \overset{\Delta}{=} \sum_{i=1}^{N} R_{ii} C_i \,, \tag{6.5}$$

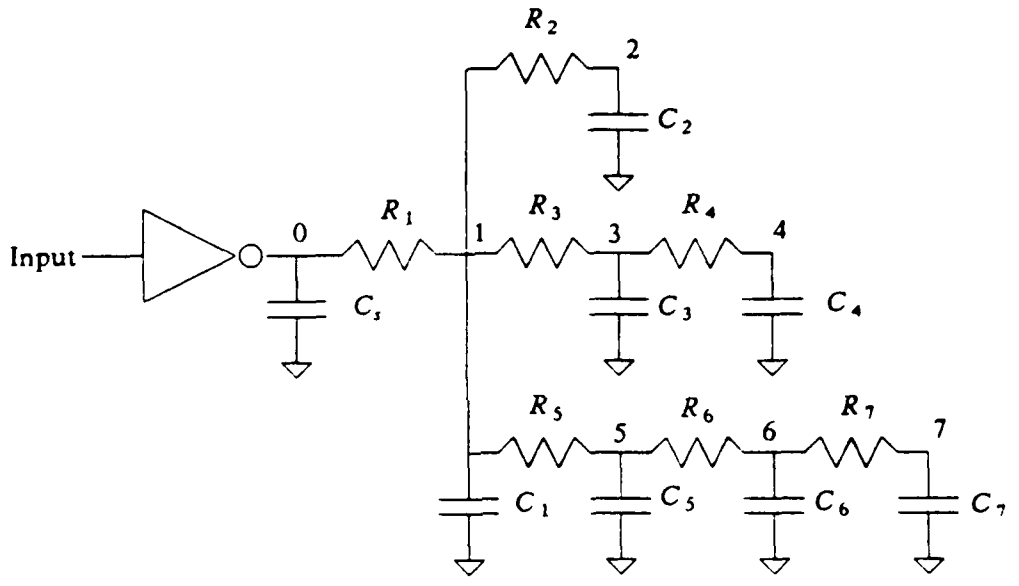
$$T_{Rk} \overset{\Delta}{=} \sum_{i=1}^{N} \frac{R_{ki}^2 C_i}{R_{kk}} \tag{6.6}$$

6.3. Node Reduction for Critical Path Timing Estimation

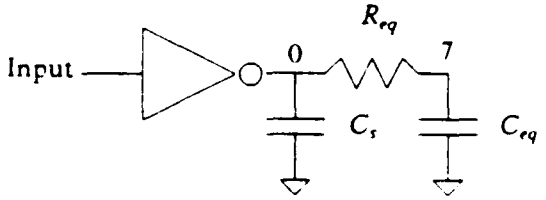
6.3.1. Problem formulation

The problem we consider here can be described as follows. Let **N** denote a driver plus an *RC* tree network with the driver input node and a set **O** of output nodes. For the timing estimation of a given output node $k \in \mathbf{O}$ we wish to replace the above network with a network $\mathbf{N}_R^I$ consisting of a driver plus a simple *RC* network with only one resistance $R_{eq(k)}^I$ between the root node and node k and one capacitance $C_{eq(k)}^I$ between node k and ground. The criterion for choosing $R_{eq(k)}^I$ and $C_{eq(k)}^I$ is that the response at node k due to a unit step input at the input of

driver in $\mathbf{N}_R^T$ should "closely" match the corresponding response in the original network **N**. Figure 6.1 shows an original *RC* tree and its reduced network used for timing estimation.



(a) An inverter driving an RC tree



(b) An inverter driving a reduced network

Figure 6.1 An example *RC* tree and its reduced network.

6.3.2. Previous approach

A waveform-bound technique has been proposed to reduce an *RC* tree in a timing estimation program called AUTODELAY [64]. The node-reduction procedures used in AUTODELAY can be summarized as follows:

(1) For a given voltage threshold V_T, the propagation delay $T_{PD(k)}$ is approximated by finding the cross point between the mean response of P-R bounds and this given threshold V_T, namely, solving the following equation for time t:

$$V_T = (v_{lk}(t) + v_{uk}(t))/2 . \tag{6.7}$$

(2) Find the equivalent single-time constant $\tau_k = R_{eq(k)}C_{eq(k)}$ for node k based on the delay $T_{PD(k)}$ by solving the following equation:

$$V_T = 1 - e^{\frac{-T_{PD(k)}}{\tau_k}} \tag{6.8}$$

(3) Calculate R_{kk}.

(4) Compute $R^I_{eq(k)}$ and $C^I_{eq(k)}$ by

$$R^I_{eq(k)} = R_{kk} . \tag{6.9}$$

$$C^I_{eq(k)} = \frac{\tau_k}{R_{kk}} . \tag{6.10}$$

It should be noted that the propagation delay of node k through the *RC* tree can also be approximated by

$$T_{PD(k)} = \frac{t_1 + t_2}{2} , \tag{6.11}$$

where t_1 and t_2 satisfy the equations $V_T = v_{lk}(t_1)$ and $V_T = v_{uk}(t_2)$. Recently, several researchers [64-67] have applied these types of methods in estimating the signal propagation delay through critical paths or in performing the node reduction of interconnect *RC* trees in the VLSI chip.

6.3.3. Our approach

The AUTODELAY approach for the node reduction of an *RC* tree is to choose $R^I_{eq(k)} = R_{kk}$ and $C^I_{eq(k)}$ to satisfy

$$R^I_{eq(k)} C^I_{eq(k)} = \tau_k \quad . \tag{6.12}$$

where $\tau_{k)}$ is computed from the P-R bounds using Equation (6.8). The motivation here is that the responses in the original and reduced networks will have approximately the same delay. However, there is still a degree of freedom in choosing individual values for $R_{eq(k)}$ and $C_{eq(k)}$ (only their product is required to be a constant). The AUTODELAY method sets $R^I_{eq(k)}$ to R_{kk} first, then computes $C^I_{eq(k)}$. If the root node in either network is driven by an ideal voltage source, then this poses no problem. However, as in the case of VLSI circuits, if the root node is driven by a nonideal source (such as an inverter element) then the source resistance and the capacitance at the root node also affect the choice of the individual values of the elements in the reduced network.

In this section we consider a new technique to compute the values of the individual elements $R^I_{eq(k)}$ and $C^I_{eq(k)}$ in the reduced network. First we compute the Elmore's time constant T_{Dk} between the root and node k in the original network. It must be noted that this computation merely involves simple multiplication and addition and is therefore much simpler than computing τ_k by Equations (6.7) and (6.8). We then require $R^I_{eq(k)}$ and $C^I_{eq(k)}$ to satisfy

$$R^I_{eq(k)} C^I_{eq(k)} = T_{Dk} \quad . \tag{6.13}$$

The motivation for such requirement is based on the following theorem.

Theorem 7.1

Consider an approximation of the waveform $v_k(t)$ at node k in the *RC* tree by the waveform $v_{Elm\,k}(t) \overset{\Delta}{=} (1 - e^{-t/T_{Dk}})$ for a step input. It can be shown that this waveform $v_{Elm\,k}(t)$ falls between P-R bounds [62,68], i.e., $v_{lk}(t) \leq v_{Elm\,k}(t) \leq v_{uk}(t)$.

Proof:

It is noted that for an *RC* tree, $T_{Rk} \leq T_{Dk} \leq T_P$, there are three cases for $v_{Elm\,k}(t) - v_k(t)$

(1) $t \leqslant T_{Dk} - T_{Rk}$:

$$v_{Elm\,k}(t) - v_{ik}(t) = 1 - e^{-t/T_{Dk}} \geqslant 0 \quad ;$$

(2) $T_{Dk} - T_{Rk} \leqslant t \leqslant T_P - T_{Rk}$:

$$v_{Elm\,k}(t) - v_{ik}(t)$$

$$= -e^{-t/T_{Dk}} + \frac{T_{Dk}}{t + T_{Rk}} \geqslant -\frac{T_{Dk}}{t + T_{Dk}} + \frac{T_{Dk}}{t + T_{Rk}} \geqslant 0 \quad (T_{Rk} \leqslant T_{Dk}) \quad ;$$

(3) $T_P - T_{Rk} \leqslant t$:

$$v_{Elm\,k}(t) - v_{ik}(t)$$

$$= -e^{-t/T_{Dk}} + \frac{T_{Dk}}{T_P} e^{(T_P - T_{Rk} - t)/T_P}$$

$$\geqslant -e^{-t/T_{Dk}} + \frac{T_{Dk}}{T_P} e^{(T_P - T_{Rk} - t)/T_{Dk}} \qquad (T_P - T_{Rk} \leqslant t)$$

$$\geqslant e^{-t/T_{Dk}} \left| -1 + \frac{T_{Dk}}{T_P} e^{(T_P - T_{Rk})/T_{Dk}} \right|$$

$$\geqslant e^{-t/T_{Dk}} \left| -1 + \frac{T_{Dk}}{T_P} \left| 1 + \frac{T_P - T_{Rk}}{T_{Dk}} \right| \right|$$

$$\geqslant e^{-t/T_{Dk}} \left| -1 + \frac{T_{Dk}}{T_P} \left| 1 + \frac{T_P - T_{Dk}}{T_{Dk}} \right| \right|$$

$$= 0 \quad ;$$

and there are two cases for $v_{ik}(t) - v_{Elm\,k}(t)$:

(1) $t \leqslant T_{Dk} - T_{Rk}$:

$$v_{ik}(t) - v_{Elm\,k}(t)$$

$$= e^{-t/T_{Dk}} - \frac{T_{Dk} - t}{T_P} \geqslant 1 - \frac{t}{T_{Dk}} - \frac{T_{Dk} - t}{T_P} \geqslant 1 - \frac{t}{T_{Dk}} - \frac{T_{Dk} - t}{T_{Dk}} = 0 \quad ;$$

$$(T_{Dk} \leqslant T_P)$$

(2) $T_{Dk} - T_{Rk} \leq t$:

$$
\begin{aligned}
& v_{ak}(t) - v_{Elm,k}(t) \\
&= e^{-t/T_{Dk}} - \frac{T_{Rk}}{TP} e^{(T_{Dk} - T_{Rk} - t)/T_{Rk}} \\
&\geq e^{-t/T_{Dk}} - \frac{T_{Rk}}{TP} e^{(T_{Dk} - T_{Rk} - t)/T_{Dk}} \qquad (T_{Dk} - T_{Rk} \leq t) \\
&\geq e^{-t/T_{Dk}} \left[1 - \frac{T_{Rk}}{TP} e^{(T_{Dk} - T_{Rk})/T_{Dk}} \right] \\
&\geq e^{-t/T_{Dk}} \left[1 - \frac{T_{Rk}}{TP} \left[1 + \frac{T_{Dk} - T_{Rk}}{T_{Dk}} \right] \right] \\
&\geq e^{-t/T_{Dk}} \left[1 - \frac{T_{Rk}}{TP} \left[1 + \frac{T_P - T_{Rk}}{T_{Rk}} \right] \right] \\
&= 0 \qquad \text{Q.E.D.}
\end{aligned}
$$

In the next section we will discuss the problem of the nonzero source resistance of the driver element in more detail and derive a further restriction on $C^{I}_{eq(k)}$ which together with Equation (6.13) will yield explicit values for $R^{I}_{eq(k)}$ and $C^{I}_{eq(k)}$.

6.3.4. Node reduction technique

Since Elmore's time constant is a good single time constant to approximate a multiple time-constant system, in our approach we first let $R^{I}_{eq(k)}$ and $C^{I}_{eq(k)}$ satisfy Equation (6.13). The choice of $R^{I}_{eq(k)}$ and $C^{I}_{eq(k)}$, however, depends strongly on the loading effect of an *RC* tree to its driver, and is not arbitrary. If $R^{I}_{eq(k)}$ is simply taken as R_{ak}, the loading effect of the calculated $C^{I}_{eq(k)}$ to driver is somewhat inaccurate. In finding an appropriate $C^{I}_{eq(k)}$, we shift the root of the *RC* tree to the input node of driver and include another constraint, namely, Elmore's time constant from input node of non-ideal driver to output node stays the same before and after the node reduction. Let R_s and C_s be on-resistance and output capacitance of

the driver, respectively. If T_{Dk}^{Io} and T_{Dk}^{Ir} are Elmore's time constants computed from the input node of the driver to output node k in the original and the reduced circuits, respectively, this constraint can be written as follows:

$$T_{Dk}^{Io} = T_{Dk}^{Ir} \quad . \tag{6.14}$$

where

$$T_{Dk}^{Io} = R_s \left(C_s + \sum_{i=1}^{N} C_i \right) + T_{Dk} \tag{6.15}$$

and

$$T_{Dk}^{Ir} = R_s \left(C_s + C_{eq(k)}^{T} \right) + T_{Dk} \tag{6.16}$$

From the above constraint and Equation (6.13), the values of $R_{eq(k)}^{T}$ and $C_{eq(k)}^{T}$ can be given by

$$C_{eq(k)}^{T} = \sum_{i=1}^{N} C_i \quad . \tag{6.17}$$

$$R_{eq(k)}^{T} = \frac{T_{Dk}}{C_{eq(k)}^{T}} \tag{6.18}$$

As an example, the following equality must hold for the circuit shown in Figure 6.1.

$$R_s \left(C_s + C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7 \right) + R_{eq(7)}^{T} C_{eq(7)}^{T}$$

$$= R_s \left(C_s + C_{eq(7)}^{T} \right) + R_{eq(7)}^{T} C_{eq(7)}^{T} \quad . \tag{6.19}$$

where R_s and C_s are the on-resistance and the output capacitance of the driver, respectively. On simplification, it is clear that $C_{eq(7)}^{T}$ has to be equal to the sum of all capacitances in the RC tree. Then from Equation (6.13) the value of $R_{eq(7)}^{T}$ can be easily determined. In this particular example, $C_{eq(7)}^{T} = C_1+C_2+C_3+C_4+C_5+C_6+C_7$ and $R_{eq} = T_{D7}/C_{eq(7)}^{T}$. It is noticed that in the constraint formulation, although the on-resistance R_s and the output capacitance C_s enter the expression, they are cancelled out in the final formulas. Therefore, this technique is independent of the on-resistance and the output capacitance of the driver.

For the purpose of comparison, a modified AUTODELAY method which utilizes the following equations to find $R^T_{eq(k)}$ and $C^T_{eq(k)}$ is also studied.

$$R^T_{eq(k)} = R_{kk} \ . \tag{6.20}$$

$$C^T_{eq(k)} = \frac{T_{Dk}}{R^T_{eq(k)}} . \tag{6.21}$$

This method is similar to the AUTODELAY method except that the equivalent time constant is T_{Dk} instead of τ_k calculated by Equation (6.8) via P-R bounds. The example circuit used in this study along with the simulation output waveforms are depicted in Figure 6.2. The percentage error in the propagation delay is -42.3 for the modified AUTODELAY method, while the percentage error is only -3.8 by using our method. These numbers indicate that our technique is substantially better than the above mentioned technique in terms of accuracy. This is particularly true when the on-resistance of driver is very high. In such case, the loading effect of capacitance to driver becomes more pronounced and cannot be neglected.

6.4. Node Reduction for Circuit Simulation

6.4.1. Problem formulation

If we want to use the node-reduction technique to reduce *RC* networks for circuit simulation, then the problem here is different from the problem previously described. The node-reduction problem for circuit simulation can be described as follows. Let **N** denote a driver plus an *RC* tree network with the driver input node and a set **O** of output nodes. We wish to replace the above tree network with a network $\mathbf{N}_R$ consisting of the same driver plus a simple *RC* network, in which only the root node and the set of nodes in **O** are involved, such that for each node $k \in \mathbf{O}$ there is a resistance $R_{eq(k)}$ between the root and node k and a capacitance $C_{eq(k)}$ between the node k and the ground node. The criterion for choosing the element values in $\mathbf{N}_R$ is that the response at node $i \in \mathbf{O}$ due to a unit step input at the input of the driver in

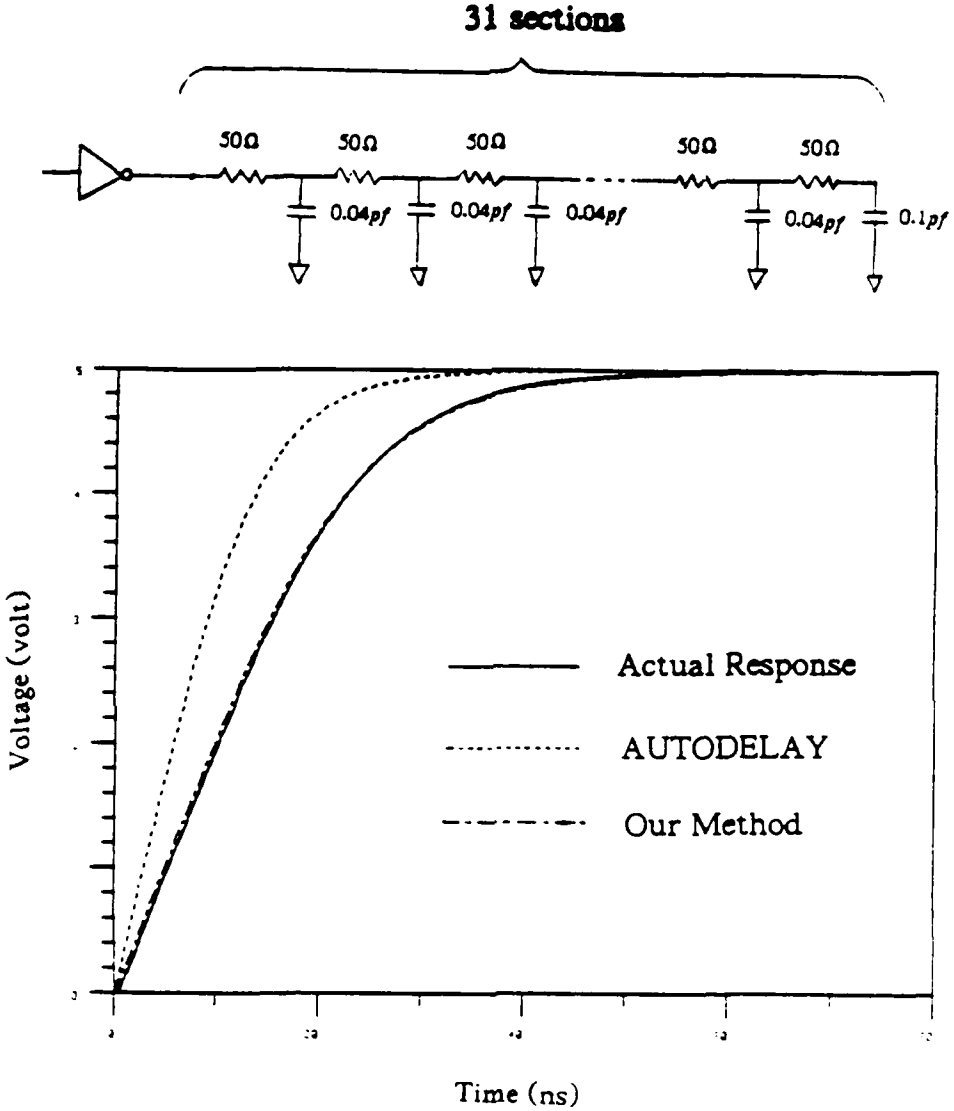


Figure 6.2 Inverter driving an RC line and its simulation output.

N_R should "closely" match the corresponding response in the original network N. Figure 6.3 shows the desired circuit schematic after the node reduction is performed on the circuit illustrated in Figure 6.1.

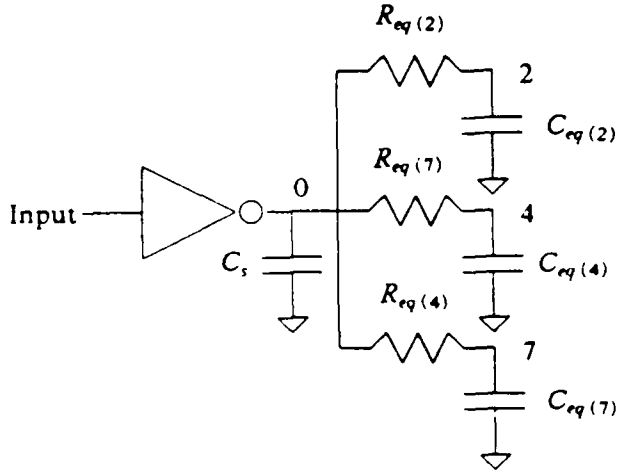


Figure 6.3 A reduced circuit for circuit simulation.

6.4.2. Heuristic algorithm for node reduction in extracted circuit output

In contrast to finding just one pair of $R^T_{eq(k)}$ and $C^T_{eq(k)}$ in the node reduction for timing estimation, our goal here is to find every pair of $R_{eq(k)}$ and $C_{eq(k)}$ for each $k \in \mathbf{O}$ which accurately approximates the delays from the input node to output nodes, regardless of electrical parameters associated with the driver. However, the method we develop is still based on the concept of node reduction for timing estimation. Two constraints on the values of $R_{eq(k)}$ and $C_{eq(k)}$ after node reduction are enforced:

(1) Elmore's time constants computed from the root are equal in the original and reduced networks, namely,

$$T_{Dk} = T^R_{Dk} \qquad \text{for } k \in \mathbf{O}. \tag{6.22}$$

It should be noted that Elmore's time constant after node reduction at node k is equal to $R_{eq(k)}C_{eq(k)}$.

(2) Elmore's time constants computed from the input of the driver are equal in the original and reduced networks, namely,

$$T_{Dk}^{I} = T_{Dk}^{IR} \qquad \text{for } k \in \mathbf{O}. \tag{6.23}$$

where

$$T_{Dk}^{I} = R_s \left(C_s + \sum_{i=1}^{N} C_i \right) + T_{Dk} \,. \tag{6.24}$$

and

$$T_{Dk}^{IR} = R_s \left(C_s + \sum_{i \in \mathbf{O}} C_{eq(i)} \right) + T_{Dk} \tag{6.25}$$

Consider an RC tree with m output nodes. From the first constraint, m constraint equations can be derived. But only one constraint equation will be obtained from the second constraint after cancellation, namely

$$\sum_{i \in \mathbf{O}} C_{eq(i)} = \sum_{i=1}^{N} C_i \tag{6.26}$$

Therefore, in this case we have to solve $2m$ variables from $m+1$ constraint equations. Except for $m=1$, the number of variables is less than the number of equations. In order to compute the exact values for $R_{eq(i)}$ and $C_{eq(i)}$, a heuristic method based on Equation (6.26) is employed to calculate m capacitances. Equation (6.22) is then used to calculate the rest of m resistances.

The heuristic node-reduction algorithm can be described as follows:

Node Reduction Algorithm

1. For each $k \in \mathbf{O}$ determine Elmore's time constant T_{Dk} between root node 0 and node k.

2. For each output node $k \in \mathbf{O}$, calculate the equivalent $R_{eq(i)}$ and $C_{eq(i)}$ as follows:

$$C_{eq(i)} = \frac{C_i}{\sum_{i \in \mathbf{O}} C_i} C_T \,. \tag{6.27}$$

$$R_{eq(k)} = \frac{T_{Dk}}{C_{eq(k)}} \quad . \tag{6.28}$$

where $C_T = \sum_{i=1}^{N} C_i$ is the total capacitance in the RC tree.

3. For each $k \in \mathbf{O}$, T_{Dk} can be used to estimate the propagation delay. Therefore, given a delay threshold, we can determine whether lumping the equivalent resistance $R_{eq(k)}$ is really necessary. We report $R_{eq(k)}$ only when the value of T_{Dk} is greater than a certain delay threshold. Otherwise, we discard the element $R_{eq(k)}$ and just merge the two nodes.

It must be noted that fan-out gate capacitances need to be added in the interconnect RC trees when applying this algorithm and should be subtracted from the output node capacitances. Therefore, the main consideration taken in step 2 is to make sure that the value of the output capacitance is greater than zero after substracting fan-out gate capacitances. The simplest distribution of C_T which meets the above requirement is shown in step 2, namely, using the output node capacitance as a weight to distribute C_T.

As an example, Figure 6.4 shows a circuit where a driver has an RC tree as a load. After applying the node-reduction algorithm, the reduced circuit is shown in Figure 6.5. Assume that the on-resistance and output capacitance of the driver are 500 Ω and 0.2 pf, respectively. The output waveforms of node 21 before and after using the node-reduction technique for a 1.66 V/ns ramp input are shown in Figure 6.6. The propagation delays and rise times for all output nodes are also listed in Table 6.1. Comparisons of output responses show reasonable agreement between the two outputs. This is because in the node-reduction process we have already considered the loading effect of each output node. Some errors produced are due to the use of a single-time-constant system approximating a multiple-time-constant system. Also the increase of R_i and C_i decreases the errors observed in the output waveforms. The reason for this is that R_i and C_i play dominant roles in determining the output waveforms when the values of R_i and C_i become larger.

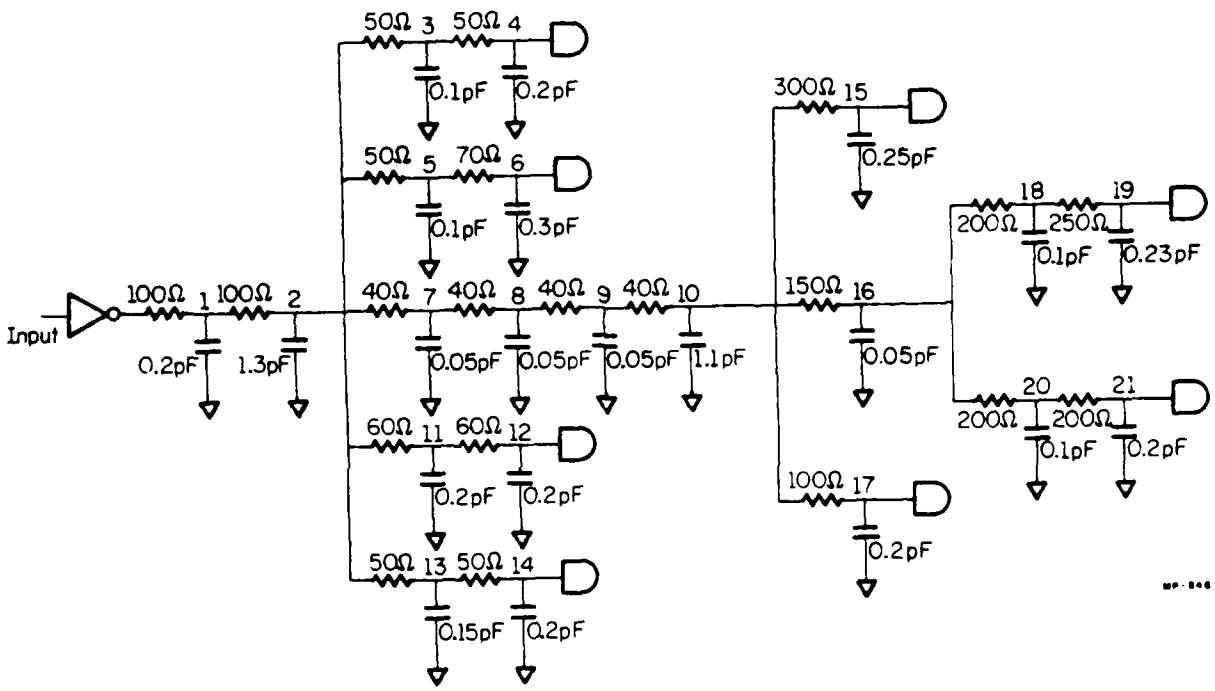


Figure 6.4 An example *RC* tree with an inverter driver.

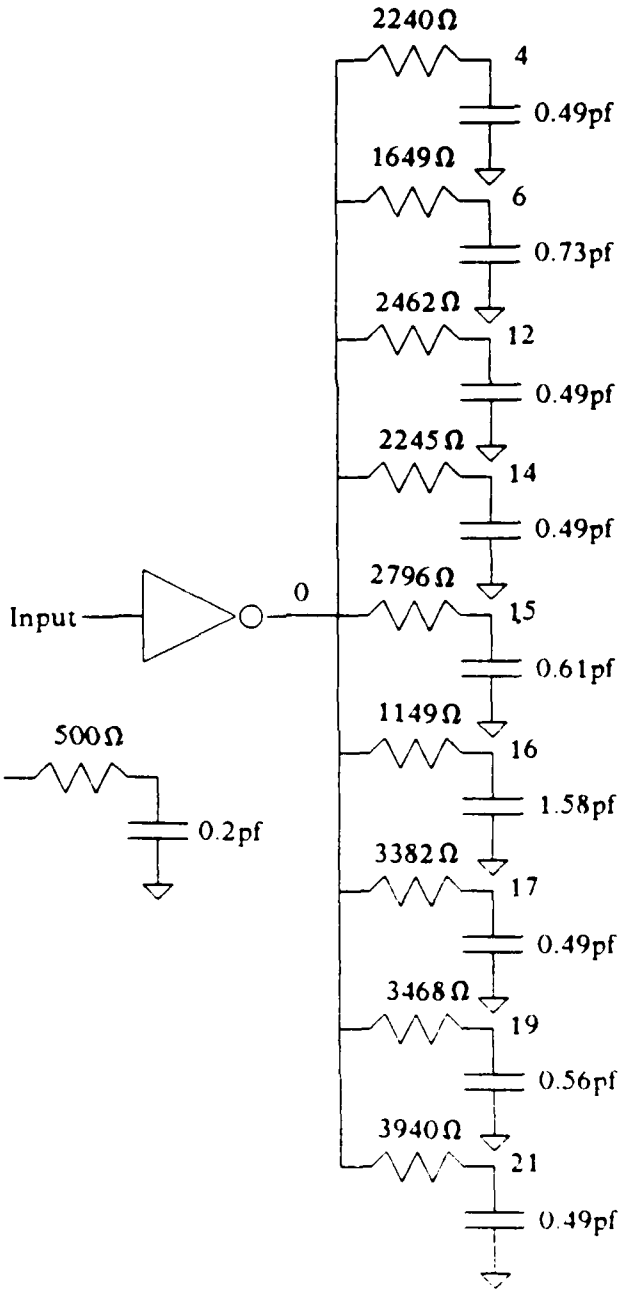


Figure 6.5 A reduced circuit corresponding to the circuit shown in Figure 6.4.

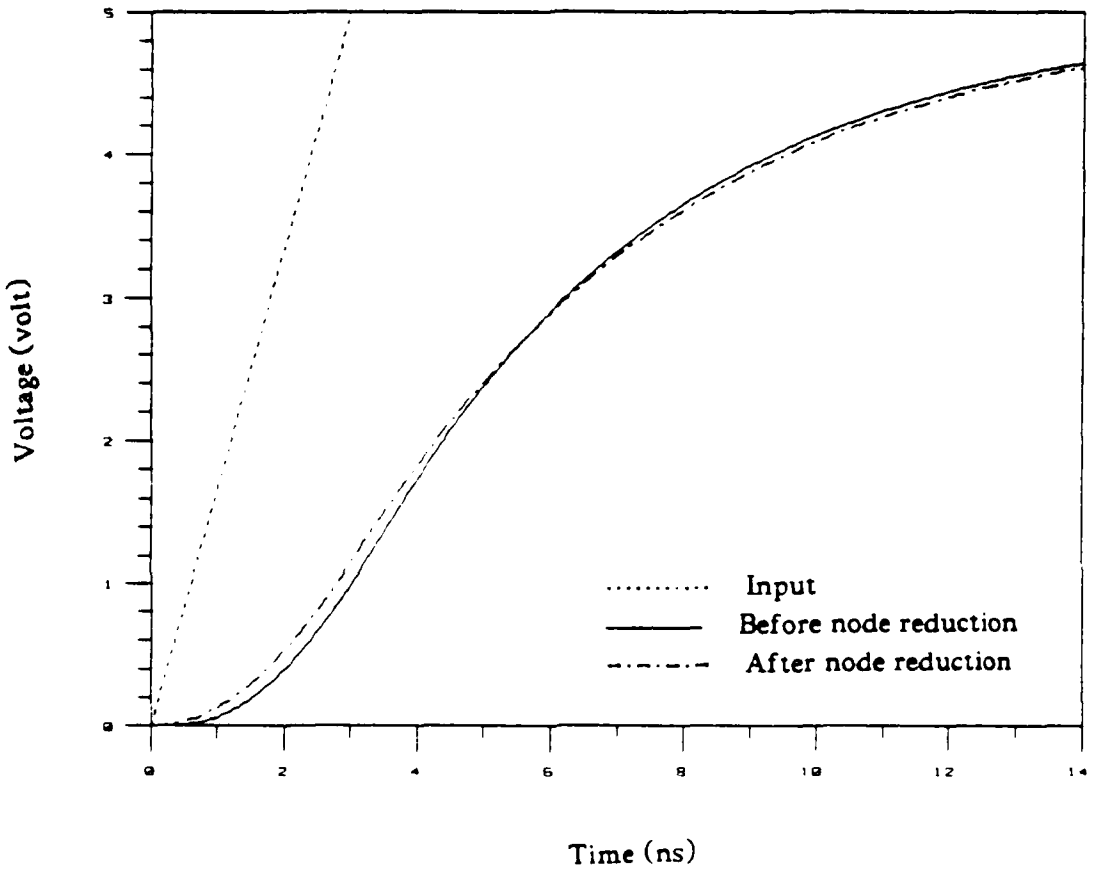


Figure 6.6 Output waveforms at node 21 before and after node reduction.

Table 6.1 Propagation and rise-time delays of the circuit shown in Fig. 6.4.

Node Number	Before node reduction		After node reduction			
	T_{PD} (ns)	T_{rise} (ns)	T_{PD} (ns)	Error (%)	T_{rise} (ns)	Error (%)
4	2.94	10.09	2.88	2.0	10.04	0.5
6	2.94	10.08	2.90	1.4	10.04	0.4
12	2.94	10.08	2.90	1.4	10.04	0.4
14	2.94	10.08	2.90	1.4	10.04	0.4
15	3.50	10.21	3.45	1.4	10.55	3.3
16	3.62	10.22	3.57	1.4	10.69	4.6
17	3.42	10.20	3.39	0.9	10.49	2.8
19	3.74	10.23	3.74	0.0	10.84	6.0
21	3.71	10.23	3.67	1.1	10.81	5.7

6.4.3. Implementation

In applying the above algorithm to circuit extraction, first of all, the circuit extractor should be able to model interconnects as *RC* networks. In our circuit extractor we have employed a two-segment, π-lumped *RC* circuit model to approximate the distributed behavior of all branches in the interconnection region. Every net is then tested to see whether further node reduction is possible, depending on the net type and connection information. Generally, if the net is connected to the pull-up node and no pass transistors are connected to it, this net can be passed to the node reduction module for further processing. In the node-reduction module, the driver node associated with each net is first identified as a root. As for the *RC* tree synthesis algorithm, the dynamic tree node allocation technique is utilized in order to reduce memory overhead and make the *RC* tree construction more flexible. Also note that during the tree construction step fan-out gate capacitances have to be estimated and added to the associated nodes of the *RC* tree in order to correctly take output capacitive loadings into account. After the *RC* tree corresponding to this root has been constructed, the depth-first search algorithm is used to find Elmore's time constant of every output node. The algorithm we use is similar to the

"TREE algorithm" presented in [62]. Finally, this information is used to produce simple lumped *RC* networks by the heuristic algorithm given in the previous section. Since the heuristic includes the filtering process, negligible resistances are always discarded and will not be reported in the extraction output.

The above algorithm is implemented in our extractor as follows:

```
Input: transistor netlist M_O and nets N_O.
Output: transistor netlist M_R and nets N_R with
        reduced RC networks.

PROCEDURE REDUCETREE(M_O, N_O);

begin
   1. From M_O find a set of pull-up nodes P.
   2. for (each node i ∈ P) do
      begin
   3.    Fetch the associated net N_i ∈ N_O of node i;
   4.    Construct the RC tree T_i from net N_i and
           set the root of tree to i;
   5.    if (no pass transistors connected to T_i) then
         begin
   6.       Add fan-out node capacitances to the
              corresponding output nodes in T_i;
   7.       Perform node reduction and filtering process;
              (Node reduction algorithm)
   8.       Update node number for transistors;
   9.       Subtract fan-out capacitances from the
              corresponding output nodes;
         end;
      end;
   10. return(M_R and N_R);
end;
```

6.5. Examples

The first example we will illustrate is an NMOS one-bit full adder. Circuit extractions were performed with and without applying the node reduction, and followed by SPICE2 [43] circuit simulations. Figure 6.7 shows the output waveforms of the sum bit before and after performing the node-reduction technique. This simulation indicates that both waveforms agree very well. However, simulation time for the circuit with node reduction is 23% less than that

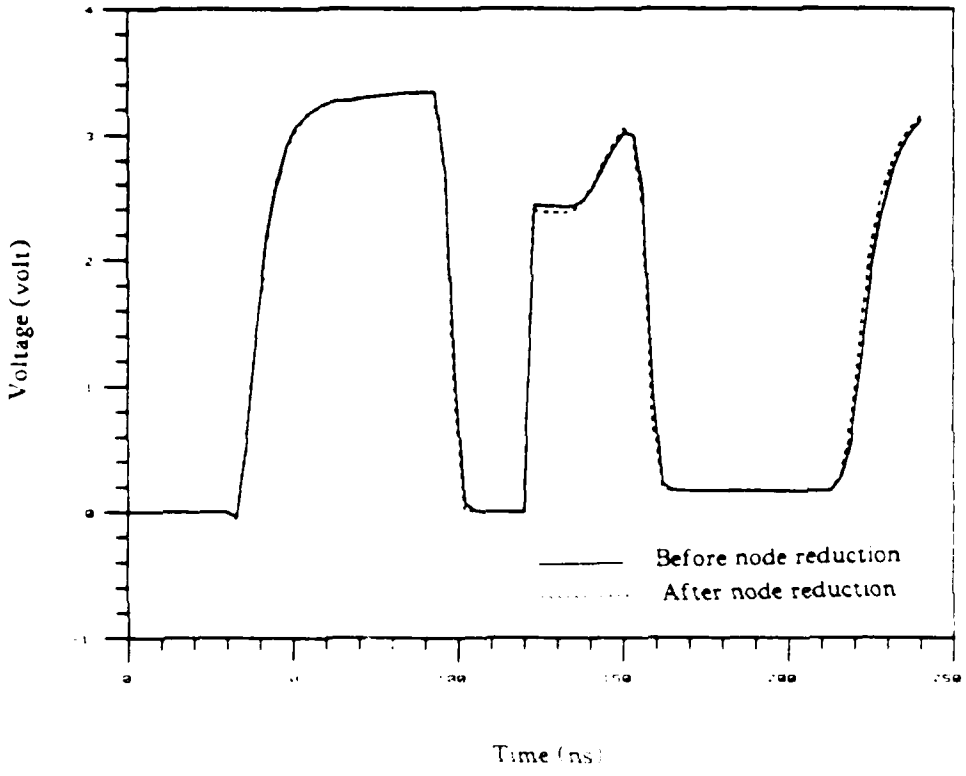


Figure 6.7 Output waveforms of the sum bit for an NMOS one-bit full adder

for the similar circuit without node reduction. In practice, the percentage of CPU time saved depends quite heavily on the number and size of the interconnect *RC* trees that can be reduced by this technique. In the worst case there could be no savings at all, i.e., the number of nodes in a circuit does not change, even after the node-reduction technique has been performed.

As another example, the layout of a CMOS PLA consisting of 44 transistors was extracted with and without the node reduction. The circuit output consisted of 81 resistors and 99 capacitors without any node reduction, while it consists of 56 resistors and 74 capacitors after node reduction. Simulation results also show good agreement between the two waveforms. In this particular example, the simulation time is reduced by 10% with the node reduction. Finally, a 4-bit NMOS full adder consisting of 82 transistors was composed from the 1-bit full adder previously considered, and was extracted. The extraction output contains 154 resistors and 190 capacitors before the node reduction is performed. However, after the node reduction only 70 resistors and 106 capacitors are left in the output. A substantial number of resistors and capacitors is reduced or filtered out in this case. The results from circuit simulations still confirm the accuracy of our proposed node-reduction technique.

6.6. Conclusions

A simple node-reduction technique for *RC* tree networks is presented in this chapter. This method computes the values of the resistances and capacitances in the reduced network using the concept of Elmore's time constant. Circuit simulations show that application of this technique reduces complicated *RC* trees into simple lumped *RC* networks without significant loss in accuracy. Therefore, this technique is useful in circuit extraction. It can be used to decrease the number of nodes in *RC* tree networks for modeling the interconnect regions in circuit layouts, thereby reducing the circuit simulation time in the next step of layout verification. A heuristic algorithm based on this technique has been implemented in our experimental circuit extractor. Simulation results show a good agreement in timing for *RC* networks obtained before and after applying this technique. Another potential application of this node-reduction technique is mentioned in [64-65], wherein every *RC* tree is simplified to a single *RC* circuit, and then delay equations are used to estimate timing through a user-specified critical path.

6.7. Remarks

The node-reduction technique can easily be adapted to *RC* mesh networks in which some resistance loops exist. In this more general case, the definition of Elmore's time constant mentioned in the first section is still valid. However, the value of R_{ii} can no longer be determined by inspection of the topology of mesh networks. Two methods can be used to calculate Elmore's time constant in a general *RC* circuit. The first method follows the definition of Elmore's time constant by finding the conductance matrix first. The conductance matrix for the *RC* mesh networks can be easily obtained by the stamp method proposed in the modified nodal analysis [69], which is widely used in circuit analysis programs. Then the resistance matrix is found by inversion of the conductance matrix, and R_{ij} is simply the ki^{th} entry of this resistance matrix [70]. Since this method involves matrix inversion, it may be very time-consuming. The second method, instead of finding R_{ij}, is to compute Elmore's time constants directly by the iterative scheme as mentioned in [62]. This method employs the concept of tree decomposition and load redistribution and solves a system of linear equations. However, no matter which method is chosen, once the value of Elmore's time constant is known, the application of our technique is fairly straightforward.

CHAPTER 7.

CONCLUSIONS

Layout verification, which checks if the design works as originally intended, is a crucial step in the final stage of a VLSI chip design. Layout design verification typically consists of three functions: (1) design-rule check (DRC), (2) functional verification, and (3) performance verification. These functions are explained below.

Design rules are specified in terms of a set of geometric constraints, which are the physical limitations imposed by the fabrication process. For example, a metal line cannot be narrower than a certain width; otherwise, it will be subjected to an electromigration reliability problem or possibly a break-up due to misalignment. Design rule checks involve intensive geometric manipulations. Violations in design rules will decrease the chip yield. Therefore, all critical design rules have to be checked before the design is sent to be fabricated.

Functional verification checks the extracted function of a layout with the intended function. Circuit extraction is first applied to obtain the transistor netlist from the layout. This extracted netlist is then checked for connectivity against the original schematic by a schematic comparison program, or an electrical rule checker can be used to detect electrical rule violations such as power-to-ground shorts. Logic or circuit simulation can also be employed to check the function of the extracted circuit. However, it is input-dependent and cannot pinpoint error locations correctly.

Performance verification checks the timing characteristics against the design specification. Circuit parameters related to propagation delays must be carefully modeled and extracted. For example, the extractor has to provide a list of transistors and interconnect parasitics, and their connectivities and associated parameters, such as channel widths and lengths, capacitance and

resistance values. This extracted information is then fed into a circuit simulator to check if the design satisfies the timing specifications.

Handcrafted and semiautomated design techniques result in error-prone layouts, which require all three types of layout verification. Recently, there is a trend toward the use of fully-automated design techniques such as silicon compilers and macrocell assemblers. These "correct-by-construction" tools are able to implement circuit layouts from a high-level description without human intervention. Although design rules and the layout function may not be checked, performance verification is still a necessity to guarantee a successful design. This is especially true in today's VLSI circuits with higher complexity and even smaller feature sizes. Propagation delays due to interconnects in a VLSI chip become almost comparable to gate delays. Hence, in order to predict the timing effect caused by interconnect parasitics, these parasitics must be directly modeled from actual physical layouts.

In this thesis, performance verification of VLSI circuits related to critical interconnect parasitics is tackled from several respects. First, two-dimensional numerical models for interconnects are studied, and are used to produce the reference values for deriving closed-form formulas. Second, in order to automate the parasitic identification process, a circuit extractor that takes a circuit layout as input and computes detailed interconnect parasitics by using the fitted formulas has been built. This step involves some critical geometric manipulations. Finally, in order to predict the parasitic effect on the timing performance of a VLSI system, a node-collapsing technique which retains the delay effect of interconnects has been studied.

The first part of this thesis deals with two-dimensional numerical models for interconnects. Based on these models, a finite-element program FEMRC has been developed and implemented on a SUN workstation. This program requires no geometric limitations in calculating capacitances and resistances due to the nature of the finite-element method. In resistance calculation, contact resistance is carefully modeled by taking a quasi-three-dimensional current flow

into account. Furthermore, only a small amount of input data has to be provided by the users because of the use of an interactive grid generator IGGI2. This feature substantially increases the usefulness of FEMRC.

In the second part of this thesis, we concentrate on developing a hierarchical parasitic and circuit extractor HPEX to deal with the increased complexity and small feature sizes of VLSI circuits. Storing layout data hierarchically, HPEX is able to handle a large volume of data as compared to flat extraction if the regularity factor is very high. HPEX also performs extraction hierarchically, thereby saving CPU time significantly by not extracting repeated cells. Main features of this program are summarized as follows:

(1) It is able to model interconnection lines as distributed lumped circuits directly from a circuit layout and automatically renumbers the generated circuit.

(2) In order to reduce computation time, analytical formulas instead of numerical methods are employed to compute interconnect resistances and capacitances which include coupling capacitances between interconnection lines.

(3) The difference in feature sizes between mask layouts and actual fabricated conductors is taken into account by a geometry preprocessing step in which a novel scanline method is used. The accuracy of the interconnect model increases if changes in feature sizes after fabrication are taken into consideration.

(4) In cell extraction, a new accurate node reduction technique is employed to facilitate design verification.

(5) The output format is exactly the SPICE input

These features demonstrate that the use of HPEX along with other simulation tools in layout verification are able to guarantee the successfulness of the first turnaround in the design cycle. Since the fabrication process, which translates a layout design into a piece of silicon, is very

costly and time-consuming, the production cost can be significantly reduced if we are able to guarantee the correctness of a VLSI design at the very beginning.

In this thesis, the signal delay affected by interconnects for layout verification is primarily emphasized. However, there are still other factors which affect the reliability or the functionality of a circuit operation. The issues of electromigration and power or ground voltage drop have recently attracted several researchers' attention [71-73] due to their important roles in the chip operation for small feature sizes. In VLSI chips power is distributed through metal lines. Current flow leads to the problems of metal migration and excessive voltage drop. Metal migration may produce a break in a metal line, causing a reliability problem. Excessive voltage drops may result in incorrect logic operation and degradation in the chip speed. Metal migration is a wear-out phenomenon in which the median time to failure (MTTF) of a metal line is inversely proportional to some power of current density. In a typical design, MTTF should pass the expected life of the system. Since the current density estimation in a metal line strongly depends on the physical layout and the circuit operation, estimations given by designers are tedious and difficult. Therefore, the need for CAD tools that are able to automate the estimation of current density in each branch of the power network is becoming increasingly apparent.

Some possible extensions of FEMRC and HPEX may cope with this problem. First, HPEX is enhanced by incorporating FEMRC to calculate resistances of irregularly shaped conductor regions. Then an *RC* network is extracted from the layout of power distribution. HPEX is also used to extract each individual cell. SPICE simulation of these cells can then be used to estimate the current loading to the extracted power *RC* network. Although the current drawn by cells is transient in time and input-dependent, it can be transformed statistically into "continuous dc electromigration equivalent values." After the current loading set is known, by SPICE simulation the current in each metal branch and the node voltages are easily computed. If the metal branch is rectangular, the estimation of current density is obvious. If the metal branch is irregular, some heuristics or FEMRC using nodal voltages as the boundary condition can be applied

to estimate the maximum current density inside this region. The current density is then checked against the specified value to see if the electromigration rule is violated. Voltage drops in power lines are also estimated at this stage. If there are some violations, a technique similar to SPIDER [72] is recommended to adjust the width of conductors.

Another possible extension of HPEX is to extract the symbolic layout from the physical layout. This symbolic layout extraction is also known as a circuit disassembler [74], which not only extracts devices and nets, but also determines the relative positions of symbolic circuit elements. This type of extraction facilitates physical layout conversion when design rules are changed by only compacting the symbolic layout. Therefore, most of the previously designed layouts can be reused even when we have different design rules.

APPENDIX A.

HPEX USER'S MANUAL

This user's manual describes how to use HPEX, a semi-interactive hierarchical circuit extractor. HPEX reads an intergrated circuit layout description in Caltech Intermediate Form (CIF 2.0) and creates five different circuit descriptions as described in Chapter 1. Comparison or simulation can then be driven from each different extracted circuit to insure the correctness of the design.

If schematics are used to produce a layout, we have to certify that the actual layout is what it is intended for. One possible technique is to use simulation as a tool for verification. However, this becomes prohibitive as the size and complexity of the circuits grow. A netlist comparison program, such as GEMINI, is therefore adopted to verify the connectivity of the extracted circuit. Two netlists are required as input to GEMINI. The first one is a transistor-level netlist from the schematics. The second one is a netlist extracted from the layout by HPEX. Netlist comparison is extremely useful in locating bugs in terms of connectivity.

If timing information is needed in verifying the timing characteristics of the circuit, the extracted circuit output has to be simulated by a timing simulator or a circuit simulator. Timing simulators such as MOSTIM can handle a large circuit by simplifying the timing model, while circuit simulators such as SPICE can only simulate a few thousand devices due to complicated device models. Typically, only the critical path of the entire chip is simulated by a circuit simulator.

In this user's manual, the input format for HPEX including some user extensions of CIF is first described. Then parameters in the process file associated with HPEX are listed and explained. Finally, by a layout example, the execution of HPEX is illustrated.

A.1. Input Format

The input to HPEX is described in a subset of CIF, namely, Manhattan geometries. Details of CIF syntax and semantics can be found in [22]. For the purpose of labeling the signal names of some particular nodes and the terminal names of a cell description, several user-extension commands on CIF have been developed. The CIF extension commands 95, 96, 90, 91, and 99 are used to name nodes. The forms of these CIF commands are as follows:

95 name x y layer;

96 name x y layer;

90 name x y layer;

91 name x y layer;

99 name x y layer;

where 95 : input node,

96 : output node,

90 : ground node,

91 : power node,

99 : terminal node,

and name : character name for signal.

These commands attach the name to the mask geometry on the specified layer crossing the point (x, y). These names may contain any ASCII character except space, tab, newline, double quotes, parenthesis, and semicolon. It should be noted that the point of a signal name must be attached to a corresponding layer described in terms of the designated integer numbers. The mapping between layers and internal integer numbers has been described in Chapter 4. In the terminal labeling, the attached point must be on the boundary of the corresponding cell.

A.2. Process File

In order to run HPEX, a process file "cmosprocess.h" has to be included in the current directory. All the necessary process parameters, which determine the values of interconnect parasitics, are contained in this file. By changing the parameters, users can easily experiment with different processes. The parameters listed in the file "cmosprocess.h" are summarized as follows:

(* Process Parameters *)

RESdiff: diffusion sheet resistance

RESpoly: polysilicon sheet resistance

RESmetal1: metal 1 sheet resistance

RESmetal2: metal 2 sheet resistance

SCRESdiff: diffusion specific contact resistivity

SCRESpoly: polysilicon specific contact resistivity

SCRESmetal: metal specific contact resistivity

THICKpoly: thickness of polysilicon

THICKmetal1: thickness of metal 1

THICKmetal2: thickness of metal 2

THICKfield: thickness of the field oxide

THICKox1: thickness of the dielectric between metal 1 and diff or poly

THICKox2: thickness of the dielectric between metal 1 and metal 2

THICKpass: thickness of passivation layer

EPSox1: permittivity of dielectric 1

EPSox2: permittivity of dielectric 2

EPSpass: permittivity of the passivation dielectric

EPSsi: permittivity of silicon

NI: intrinsic carrier concentration of the semiconductor 1.45*10**10 /cm**3 at 300k

KEQN: averaging factor for junction capacitance calculation for NMOS devices

KEQP: averaging factor for junction capacitance calculation for PMOS devices

NDN: drain, source N doping for NMOS devices

NDP: drain, source P doping for PMOS devices

NSUB: n-type substrate doping

NPWELL: p-well doping

TOX: oxide thickness

XJP: metallurgical junction depth for P diffusion

XJN: metallurgical junction depth for N diffusion

LDP: P type lateral diffusion

LDN: N type lateral diffusion

VT: thermal voltage

Q: electronic charge

CJOP: zero-bias bulk junction bottom cap. per unit area for P diff

CJON: zero-bias bulk junction bottom cap. per unit area for N diff

COX: gate oxide cap. per unit area

PBP: built-in junction potential for P diff

PBN: built-in junction potential for N diff

CGSOP: gate-source overlap cap. per meter channel width for PMOS devices

CGSON: gate-source overlap cap. per meter channel width for NMOS devices

Rth: lower threshold for resistance

Rthseg: higher threshold for resistance

Cselfth: threshold for self-capacitance

Ccoupth: threshold for coupling capacitance

NETGAP: threshold for the net distance - used in coupling capacitance calculation

BRANCHGAP: threshold for the branch distance - used in coupling capacitance calculation

HoriScale: scaling factor for the layout - only horizontal

DELAYTH: threshold for the delay used in the node reduction

GLITCHTH: threshold for voltage glitch - for filtering negligible effect of coupling capacitance

Cnj: capacitance per unit area for N diff junctions

Cnjp: capacitance per unit length for N diff junctions

Cpj: capacitance per unit area for P diff junctions

Cpjp: capacitance per unit length for P diff junctions

Cm2m1: capacitance per unit area between metal1 and metal 2

Cm2p: capacitance per unit area between metal 2 and poly

Cm1p: capacitance per unit area between metal 1 and poly

Cm1d: capacitance per unit area between metal 1 and diffusion

Cm2of: capacitance per unit area between metal 2 and field oxide

Cm1of: capacitance per unit area between metal 1 and field oxide

Cpof: capacitance per unit area between poly and field oxide

featuresized: feature size threshold for diffusion

featuresizep: feature size threshold for poly

featuresizem1: feature size threshold for metal1

featuresizem2: feature size threshold for metal 2

offset[1..nmsklvls]: offset distance for each mask level

(* END OF PROCESS PARAMETERS *)

A.3. Running HPEX

To run HPEX the command line would look as follows:

hpex [- [w] [o] [h] [e]] ciffile [outfile] [logfile]

HPEX gets command line arguments for program use. The four fields after **hpex** are defined as

{options} : A minus sign, followed by the letter *w*, or the letter *o*, or the letter *h* or the letter

e, or any combination of these.

-w

turn off all warnings. The default is to send warnings to the log file.

-o

cause tree optimization to be performed. If omitted, the default is not to perform tree optimization.

-h

turn on a hierarchical extraction. The default is to perform a flat extraction.

-e

cause nonblank characters following the CIF End command to generate a warning only. The default is to generate an error.

{ciffile} : The name of the CIF file to be used as input to the extraction program.

{outfile} : The name of the output file to which the extracted information is to be written. If this field is omitted, the default is to send the listing information to the file 'ciffile.out.'

{logfile} : The name of the log file to which the listing information is to be written. If this field is omitted, the default is to send the listing information to the file 'ciffile.log.'

A.4. Example

The CIF file "inv.cif" of a CMOS inverter shown in Figure A.1 is described as follows:

```
DS 10 1 2;
9 inv1;
L CW;
   B 4800 9000 -3600 -3900;
L CM;
   B 4800 1500 -3600 6450;
   B 1800 2400 -3600 2700;
   B 2100 1500 -3450 750;
```

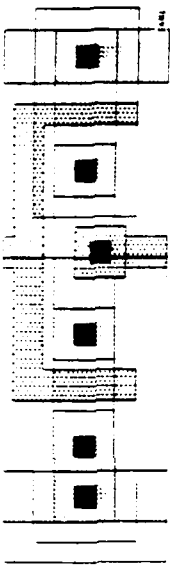


Figure A.1 An example layout of a CMOS inverter.

```
    B 1800 2400 -3600 -1200;
    B 1800 1800 -3600 -4800;
    B 4800 1500 -3600 -6450;
L CP;
    B 3600 600 -3900 4800;
    B 900 3300 -5250 2850;
    B 1200 900 -5400 750;
    B 900 3000 -5250 -1200;
    B 1500 300 -3150 1350;
    B 2700 900 -2550 750;
    B 1500 300 -3150 150;
    B 3600 900 -3900 -3150;
L CD;
    B 1800 4800 -3600 4800;
    B 1800 6300 -3600 -4050;
L CC;
    B 600 600 -3600 6450;
    B 600 600 -3600 3150;
    B 600 600 -3600 -1650;
    B 600 600 -3600 -4950;
    B 600 600 -3600 -6450;
```

```
    B 600 600 -3150 750;
L CS;
    B 3000 6000 -3600 4800;
    B 3000 2100 -3600 -6750;
99 lgnd -6000 -6600 5;
99 rgnd -1200 -6600 5;
99 inp -6000 600 3;
99 out -1200 600 3;
99 lvdd -6000 6300 5;
99 rvdd -1200 6300 5;
DF;
C 10 T 0,0;
End
```

A typical run for HPEX is to type the following command line:

```
hpex -h inv.cif inv.out .
```

Following this, several interactive questions have to be answered by the user to complete the input session. In this example, the following questions appear on the terminal screen sequentially:

Which file do you like to generate?

(1) MOSTIM input file

(2) SPICE input file

(3) GEMINI inputfile

(4) Logic block description

Enter choice : 2

Do you like to use measured parameters in capacitance computation? [n] *CR*

Do you like to calculate coupling capacitances? [n] y *CR*

Do you like to perform node reduction? [n] *CR*

Do you like to generate SPICE model cards? [n] *CR*

It should be noted that the answer "no" is defaulted to each question when we hit the carriage return. After the execution is completed, the output file "inv.out" looks as follows:

```
* INPUT FILE : inv.cif
* EXTRACTION DATE : 10 Aug 87  TIME : 15:33:31
```

```
* INPUT DECK FOR SPICE
.subckt  inv1  13 12 16 17 9 22
*rgnd  : 13
*lgnd  : 12
*rvdd  : 16
*lvdd  : 17
*out  : 9
*inp  : 22
*total no. of transistors :        2
* NET5
r0001     20    22   2.500e+01
r0002     20    19   1.587e+02
r0003     18    20   1.889e+02
c0001     18     0  0.00351pf
c0002     22     0  0.00052pf
c0003     20     0  0.00693pf
c0004     19     0  0.00364pf
* NET4
r0004     15    16   9.600e-02
r0005     15    17   9.600e-02
r0006     14    15   5.857e+00
c0005     14     0  0.00042pf
c0006     17     0  0.00167pf
c0007     16     0  0.00167pf
c0008     15     0  0.00375pf
* NET3
r0007     11    13   9.600e-02
r0008     11    12   9.600e-02
r0009     10    11   1.004e+01
r0010     10    11   1.006e+01
c0009     10     0  0.00265pf
c0010     12     0  0.00167pf
c0011     13     0  0.00167pf
c0012     11     0  0.00598pf
* NET2
r0011      8     7   5.935e+00
r0012      8     9   7.002e+01
r0013      6     8   5.101e+00
c0013      6     0  0.00250pf
c0014      9     0  0.00297pf
c0015      7     0  0.00250pf
c0016      8     0  0.00798pf
m1      7    18    14  1  pdev   l= 3.00u  w= 9.00u
+ ad= 94.50p  as= 94.50p
m2     10    19     6  0  ndev   l= 4.50u  w= 9.00u
+ ad= 162.00p  as= 81.00p
.ends  inv1
x1 12 10 13 11 9 8  inv1
*Readin time                :      0.42
*Geometric extraction time  :      1.13
*Total job time             :      1.55
```

In the output file, the user-specified node names are listed with their corresponding integer node numbers. This information will help users to identify *i/o* node numbers in the simulation phase. Since the output file is SPICE-input compatible, it is ready for simulation except that users have to supply the input waveforms.

REFERENCES

[1] D. E. Thomas, "The automatic synthesis of digital systems," *Proc. IEEE*, vol. 69, no. 10, pp. 1200-1211, October 1981.

[2] A. Gupta, "ACE: A circuit extractor," *Proc. 20th Design Automation Conf.*, pp. 721-725, 1983.

[3] S. P. McCormick, "EXCL : A circuit extractor for IC designs," *Proc. 21st Design Automation Conf.*, pp. 624-628, 1984.

[4] W. S. Scott and J. K. Ousterhout, "Magic's circuit extractor," *Proc. 22nd Design Automation Conf.*, pp. 286-292, 1985.

[5] J. B. Bastian, M. Ellement, P. J. Fowler, C. E. Huang, and L. P. McNamee, "Symbolic parasitic extractor for circuit simulation (SPECS)," *Proc. 20th Design Automation Conf.*, pp. 346-352, 1983.

[6] T. Mitsuhashi, T. Chiba, and M. Takashima, "An integrated mask artwork analysis system," *Proc. 17th Design Automation Conf.*, pp. 277-284, 1980.

[7] P. Losleben and K. Thompson, "Topological analysis for VLSI circuits," *Proc. 16th Design Automation Conf.*, pp. 461-473, 1979.

[8] H. S. Baird and Y. E. Cho, "An artwork design verification system," *Proc. 12th Design Automation Conf.*, pp. 414-420, 1975.

[9] B. T. Preas, B. W. Lindsay, and C. W. Gwyn, "Automatic circuit analysis on mask information," *Proc. 13th Design Automation Conf.*, pp. 309-317, 1976.

[10] T. Akino, M. Shimode, Y. Kurashige, and T. Negishi, "Circuit simulation and timing verification based on MOS/LSI information," *Proc. 16th Design Automation Conf.*, pp. 85-94, 1979.

[11] C. Ebeling and O. Zajicek, "Validating VLSI circuit layouts by wirelist comparison," *Digest of Technical Papers, ICCAD*, pp. 172-175, 1983.

[12] K. Hirakawa, N. Shiraki, and M. Muraoka, "Logic simulation for LSI," *Proc. 19th Design Automation Conf.*, pp. 755-761, 1982.

[13] L. W. Nagel, "SPICE2 : A computer program to simulate semiconductor circuits," *Electronics Research Report #ERL-M520*, May 1975.

[14] V. B. Rao, "Switch-level timing simulation of MOS VLSI circuits," Ph.D. dissertation, University of Illinois, Urbana, Illinois, 1985.

[15] K. C. Saraswat and F. Mohammadi, "Effect of scaling of interconnections on the time delay of VLSI circuits," *IEEE J. Solid-State Circuits*, vol. SC-17, pp. 275-280, 1982.

[16] H. B. Bakoglu and J. D. Meindl, "Optimal interconnection circuits for VLSI," *IEEE Trans. Electron Devices*, vol. ED-32, pp. 903-909, 1985.

[17] H. Yuan, Y. Lin, and S. Chiang, "Properties of interconnection on silicon, sapphire and semi-insulating gallium arsenide substrates," *IEEE Trans. Electron Devices*, vol. ED-29, pp. 639-644, 1982.

[18] M. Trick, A. J. Strojwas, and S. W. Director, "Fast RC simulation for VLSI interconnect," *Digest of Technical Papers, ICCAD*, pp. 178-179, 1983.

[19] S. Mori, I. Suwa, and J. Wilmore, "Hierarchical capacitance extraction in an IC artwork verification system," *Digest of Technical Papers, ICCAD*, pp. 266-268, 1984.

[20] S. Mori and J. Wilmore, "Resistance extraction in a hierarchical IC artwork verification system," *Digest of Technical Papers, ICCAD*, pp. 196-198, 1985.

[21] S. R. Nassif, A. J. Strojwas, and S. W. Director, "FABRICS II: A statistical process simulator of the IC fabrication process," *Proc. ICCC 82*, pp. 298-301, 1982.

[22] C. A. Mead and L. A. Conway, *Introduction to VLSI System*, Reading, MA: Addison-Wesley, 1980.

[23] Y. Okamura, Y. Muraishi, T. Sato, and Y. Ikemoto, "LAS: Layout analysis system with new approach," *Proc. ICCC*, pp. 308-311, 1982.

[24] C. M. Sakkas, "Potential distribution and multi-terminal DC resistance computations for LSI technology," *IBM J. Res. Develop.*, vol. 23, no. 6, November 1979.

[25] B. R. Chawla and H. K. Gummel, "A boudary technique for calculation of distributed resistance," *IEEE Trans. Electron Devices*, vol. ED-17, no. 10, pp. 915-925, October 1970.

[26] T. Mitsuhashi and K. Yoshida, "A resistance calculation algorithm and its appication to circuit extraction," *IEEE Trans Computer-Aided Design*, vol. CAD-6, no. 3, pp. 337-345, May 1987.

[27] C. P. Yuan, "Modeling and extraction of interconnect parameters in very-large-scale integrated circuits," Ph.D. dissertation, University of Illinois, Urbana, Illinois, 1983.

[28] A. E. Ruehli and P. A. Brennan, "Efficient capacitance calculations for three-dimensional multiconductor systems," *IEEE Trans. Microwave Theory Tech*., vol. MTT-21, no. 11, pp. 76-82, February 1973.

[29] A. Farrar and A. T. Adams, "Multilayer microstrip transmission lines," *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-22, no. 10, pp. 889-891, Octobeoberr 1974.

[30] W. H. Loh, S. E. Swirhun, T. A. Schreyer, R. M. Swanson, and K. C. Saraswat, "Modeling and measurement of contact resistances," *IEEE Trans. Electron Devices*, vol. ED-34, no. 3, pp. 512-524, March 1987.

[31] M. R. Pinto, C. S. Rafferty, H. R. Yeager, and R. W. Dutton, "PISCES-IIB supplementary report," *Stanford University*, 1985.

[32] C. P. Yuan and T. N. Trick, "Calculation of capacitance in VLSI circuits," *Digest of Technical Papers, ICCAD*, pp. 263-265, 1984.

[33] W. T. Weeks, "Calculation of coefficient of capacitance of multiconductor transmission lines in the presence of a dielectric interface," *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-18, pp. 35-43, 1970.

[34] P. E. Cottrell and E. M. Buturla, "VLSI wiring capacitance," *IBM J. Res. Develop.*, vol. 29, no. 3, May 1985.

[35] T. Sakurai and T.Tamaru, "Simple formulas for two- and three-dimensional capacitances," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 183-185, February 1983.

[36] P. M. Hall, "Resistance calculations for thin film patterns," *Thin Solid Films*, vol. 1, pp. 277-295, 1967/68.

[37] Z.-Q. Ning, P. M. Dewilde, and F. L. Neerhoff, "Capacitance coefficients for VLSI multilevel metallization lines," *IEEE Trans. Electron Devices*, vol. ED-34, no. 3, pp. 644-649, March 1987.

[38] W. H. Chang, "Analytic IC metal-line capacitance formulas," *IEEE Trans. Microwave Theory and Tech.*, pp. 608-611, September 1976.

[39] M. I. Elmasry, "Capacitance calculation in MOSFET VLSI," *IEEE Electron Device Lett.*, vol. EDL-3, no. 1, pp. 6-7, January 1982.

[40] 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 Lett.*, vol. EDL-3, no. 12, pp. 391-393, December 1982.

[41] H. B. Palmer, "The capacitance of a parallel-plate capacitor by the Schwartz-Christoffel transformation," *Trans. AIEE*, vol. vol. 56, p. 363, March 1937.

[42] K. C. Gupta, R. Garg, and I. J. Bahl, *Microstrip and Slotlines*, Dedham, MA: Artech, 1979.

[43] M. Zahn, *Electromagnetic Field Theory: A Problem Solving Approach*, New York: Wiley, 1979.

[44] T. Sakurai, "Approximation of wiring delay in MOSFET LSI," *IEEE J. Solid-State Circuits*, vol. SC-18, pp. 418-426, August 1983.

[45] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. Ass. Comput. Mach.*, vol. 18, no. 9, pp. 509-517, September 1975.

[46] J. B. Rosenberg, "Geographical data structure compared: A study of data structures supporting region queries," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, no. 1, pp. 53-67, January 1985.

[47] F. P. Preparata and M. I. Shamos, *Computational Geometry*, New York: Springer-Verlag, 1985.

[48] K. Komatsu and M. Suzuki, "The outline procedure in pattern data preparation for vector-scan electron-beam lithography," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 145-151, January 1987.

[49] M. Sato, M. Tachibana, and T. Ohtsuki, "An algorithm for resizing polygonal regions and its applications to LSI mask pattern design," *Electronics and Communications in Japan*, vol. 67-C, no. 4, pp. 93-101, 1984.

[50] J. K. Ousterhout, "The user interface and implementation of an IC layout editor," *IEEE Trans. Computer Aided Design*, vol. CAD-3, no. 3, pp. 242-249, July 1984.

[51] C. Nessen, "Hierarchical design methodologies and tools for VLSI chips," *Proc. IEEE*, vol. 71, no. 1, pp. 66-75, January 1983.

[52] G. M. Tarolli and W. J. Herman, "Hierarchical circuit extraction with detailed parasitic capacitance," *Proc. 20th Design Automation Conf.*, pp. 337-345, 1983.

[53] L. V. Corbin, "Custom VLSI electrical rule checking in an intelligent terminal," *Proc. 18th Design Automation Conf.*, pp. 693-701, 1981.

[54] M. E. Newell and D. T. Fitzpatrick, "Exploitation of hierarchy in analyses of integrated circuit artwork," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, no. 4, pp. 192-200, October 1982.

[55] M. A. Shand, "Hierarchical VLSI artwork analysis," *Proc. International Conf. on VLSI*, pp. 419-428, 1985.

[56] A. Bootehsaz and R. A. Cottrell, "A technology independent approach to hierarchical IC layout extraction," *Proc. 23rd Design Automation Conf.*, pp. 425-431, June, 1986.

[57] M. Horowitz and R. W. Dutton, "Resistance extraction from mask layout data," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, no. 3, pp. 145-150, July 1983.

[58] L. K. Scheffer and R. Soetarman, "Hierarchical analysis of IC artwork with user-defined rules," *IEEE Design and Test*, pp. 66-74, February 1986.

[59] J. K. Ousterhout, G. T. Hamachi, R. N. Mayo, W. S. Scott, and G. S. Taylor, "The Magic VLSI layout system," *IEEE Design and Test*, pp. 19-30, February 1985.

[60] J. K. Ousterhout, "Corner stitching: A data-structuring technique for VLSI layout tools," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, no. 1, pp. 87-100, January 1984.

[61] P. Penfield and J. Rubinstein, "Signal delay in RC tree networks," *Proc. 18th Design Automation Conf.*, pp. 613-617, 1981.

[62] T.-M. Lin and C. A. Mead, "Signal delay in general RC networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, no. 4, pp. 331-349, October 1984.

[63] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, pp. 55-63, January 1948.

[64] R. Putatunda, "AUTODELAY: A program for automatic calculation of delay in LSI/VLSI chips," *Proc. 19th Design Automation Conf.*, pp. 616-621., 1982.

[65] R. Putatunda, "AUTODELAY: A second-generation automatic delay calculation program for LSI/VLSI chips," *Digest of Technical Papers, ICCAD 84*, pp. 188-190, 1984.

[66] E. Tamura, K. Ogawa, and T. Nakano, "Path delay analysis for hierarchical building block layout system," *Proc. 20th Design Automation Conf.*, pp. 403-410, 1983.

[67] N. Jouppi, "Timing analysis for NMOS VLSI," *Proc. 20th Design Automation Conf.*, pp. 411-418, 1983.

[68] J. L. Wyatt, C. A. Zukowski, and P. Penfield, Jr., "Step response bounds for systems described by M-matrices, with application to timing analysis of digital MOS circuits," *Proc. 24th Conf. on Decision and Control*, pp. 1552-1557, December 1985.

[69] C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, vol. CAS-22, no. 6, pp. 504-509, June 1975.

[70] J. L. Wyatt, "Signal delay in RC mesh networks," *IEEE Trans. Circuits and Systems*, vol. CAS-32, no. 5, pp. 507-510, May 1985.

[71] W. S. Song and L. A. Glasser, "Power distribution techniques for VLSI circuits," *Conference on Advanced Research in VLSI, M.I.T.*, pp. 45-52, 1984.

[72] J. E. Hall, D. E. Hocevar, P. Yang, and M. J. McGraw, "SPIDER - A CAD system for checking current density and voltage drop in VLSI metallization patterns," *Digest of Technical Papers, ICCAD 86*, pp. 278-281, 1986.

[73] S. Chowdhury and M. A. Breuer, "The construction of minimal area power and ground nets for VLSI circuits," *Proc. 22nd Design Automation Conf.*, pp. 794-797, 1985.

[74] B. Lin and A. R. Newton, "KAHLUA: A hierarchical circuit disassembler," *Proc. 24th Design Automation Conf.*, pp. 311-317, 1987.

VITA

Shun-Lin Su was born in Hsinchu, Taiwan, Republic of China on January 11, 1956. He received a B.S. degree in Electrical Engineering from National Taiwan University, Taipei, R.O.C. in June 1979. He came to the University of Illinois in August 1981 and received his M.S. degree in Electrical Engineering in May 1983. Since August 1981, he worked as a research assistant at the Coordinated Science Laboratory and a teaching assistant with the Department of Electrical and Computer Engineering at the University of Illinois. His research interests are in the area of computer-aided design, VLSI design, semiconductor device modeling and silicon compilation.

END

12-87

DTIC