

Copyright © 1992, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**PERFORMANCE-CONSTRAINED PHYSICAL  
DESIGN OF ANALOG AND MIXED  
ANALOG/DIGITAL CIRCUITS**

by

Umakanta Choudhury

Memorandum No. UCB/ERL M92/38

16 April 1992

111  
COVER PAGE

**PERFORMANCE-CONSTRAINED PHYSICAL  
DESIGN OF ANALOG AND MIXED  
ANALOG/DIGITAL CIRCUITS**

by

Umakanta Choudhury

Memorandum No. UCB/ERL M92/38

16 April 1992

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering  
University of California, Berkeley

94720

TITLE PAGE

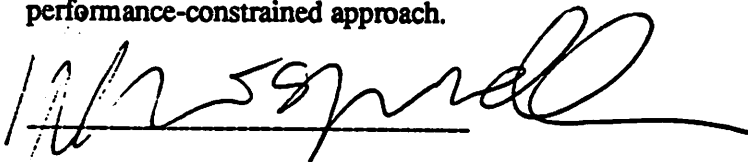
# Performance-Constrained Physical Design of Analog and Mixed Analog/Digital Circuits

Umakanta Choudhury

Department of Electrical Engineering and Computer Sciences

## Abstract

Analog design is often the bottleneck in the design of mixed analog/digital circuits. The computer-aided design (CAD) tools available for digital circuits cannot be used for the design of analog and general mixed analog/digital circuits, as such designs involve additional constraints associated with the analog circuits and interaction between analog and digital circuits. Layout design forms a critical part of the design of analog circuits, and manual layout design is very error-prone and time consuming, since the performance of the circuits can be sensitive to stray effects in the layout. A performance-constrained approach towards automatic layout design of analog and mixed analog/digital circuits is proposed in this thesis. The proposed approach involves generating a set of constraints on the parasitics from a set of specified performance constraints while trying to provide maximum flexibility to the layout tools. Driving the layout tools by these parasitic constraints reduces the need for further expensive layout iterations and in some cases may even eliminate them. A parasitic constraint generator, and a constraint-driven router have been presented. A model generator for interconnect capacitances has also been developed which generates analytical models by solving Laplace's equation and using a partial knowledge of flux components associated with the configurations of interest. These models are being used during constraint-driven layout design. Results for several test circuits have been presented to illustrate the usefulness and feasibility of the performance-constrained approach.



Prof. Alberto Sangiovanni-Vincentelli

Thesis Committee Chairman

## Acknowledgements

I am grateful to Prof. Alberto Sangiovanni Vincentelli, my research adviser, for his guidance and inspiration during the course of this doctoral work. I am thankful to him for helping me formalize many concepts and providing me with the right perspectives at several stages of this work.

I would also like to thank Prof. Brayton for serving in my qualifying and thesis committees. I thank Prof. Gray for many useful discussions regarding my project and for serving in my thesis committee. I also thank Prof. Ole Hald and Prof. James Sethian of Mathematics Department for serving in my qualifying and thesis committees respectively.

One of the motivations for pursuing this doctoral work came during my summer stay in H.P. Santa Rosa where I worked on parasitic extraction for high-frequency circuits. I thank Dr. Bruce Donecker and Dr. Gary Homeland of H.P. Santa Rosa for their guidance in that project.

Many senior colleagues provided me with technical advice and help during the initial phase of my work in the CAD group. I thank Dr. Jyuo-Min Shyu, Dr. David Riley and Dr. Tom Quarles for their help regarding sensitivity analysis in SPICE. I also thank Dr. Ken Kundert for many useful discussions on circuit simulation.

I would also like to thank colleagues in CAD group who have worked with me on projects related to my dissertation. I thank Mr. Enrico Malavasi and Mr. Edardo Charbon for working with me on constraint-driven area routing and placement. I also thank Mr. Nick Weiner for many useful discussions on analog routing, and Mr. Gani Jusuf for providing me with test circuits.

I am thankful to Jaijeet RoyChowdhury, Mani Srivastava and K.J. Singh for being good friends. I also thank Flora Oviedo and Elise Mills for being always helpful in administrative matters and Brad Krebbs for his help when I had computing problems.

Last but not least, I thank my family members in India and my wife Seema for their encouragement during the course of this work.

This research has been sponsored by DARPA under grant N00039-87-C-0182, MICRO, Harris, HP, Rockwell and AT&T.

# Contents

<b>List of Figures</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Physical Design of Mixed Analog/Digital Systems . . . . .	3
1.2 A Performance-Constrained Approach towards physical design . . . . .	5
1.3 Organization of the thesis . . . . .	8
<b>2 Thesis Background</b>	<b>10</b>
2.1 Performance Functions . . . . .	10
2.2 Performance Sensitivities . . . . .	11
2.3 Interconnect Parasitics . . . . .	15
2.3.1 Resistances . . . . .	16
2.3.2 Capacitances . . . . .	16
2.3.3 Inductances . . . . .	19
2.4 Automatic Physical Design . . . . .	20
2.4.1 Placement . . . . .	21
2.4.2 Routing . . . . .	23
2.4.3 Spacing/Compaction . . . . .	25
<b>3 Previous Work in Analog Design Automation</b>	<b>27</b>
3.1 Previous work in Circuit Synthesis . . . . .	27
3.1.1 Synthesis systems . . . . .	28
3.1.2 General-purpose Circuit Optimizers . . . . .	32
3.2 Previous Work in Layout Synthesis . . . . .	33
3.2.1 Special-purpose Layout Systems . . . . .	34
3.2.2 General-purpose Layout Systems . . . . .	35
3.2.3 Layout Tools . . . . .	40
3.3 Common Limitations of Previous Approaches in Analog Layout Design Au- tomation . . . . .	42
<b>4 Parasitic Constraint Generation</b>	<b>43</b>
4.1 Notation . . . . .	43
4.2 Performance Constraints . . . . .	45
4.2.1 Nonprecision Specifications . . . . .	46

4.2.2	Precision Specifications . . . . .	46
4.3	The parasitic constraint-generation problem . . . . .	50
4.4	Modeling Performance Constraints . . . . .	51
4.5	Matching Constraints on Parasitics . . . . .	53
4.6	Bounding Constraints on Parasitics . . . . .	55
4.6.1	Limits for Parasitics . . . . .	55
4.6.2	Selecting the Critical Parasitics . . . . .	55
4.6.3	Modeling Layout Flexibility . . . . .	57
4.6.4	The Main Algorithm . . . . .	59
4.7	PARCAR . . . . .	62
4.7.1	Program Input . . . . .	62
4.7.2	Program Operation . . . . .	63
4.8	Results . . . . .	65
4.8.1	SC Filter Example . . . . .	65
4.8.2	CMOS Opamp Example . . . . .	68
4.9	Summary . . . . .	71
4.10	Appendix: Worst-Case Sensitivity with Process Variations . . . . .	72
4.11	Appendix: Estimation of Maximum and Minimum Capacitances . . . . .	77
4.11.1	Horizontal Segments . . . . .	79
4.11.2	Vertical Segments . . . . .	81
4.11.3	Crossing Segments . . . . .	86
4.12	Appendix: Shielding Decision . . . . .	88
<b>5</b>	<b>Constraint-driven layout design</b> . . . . .	<b>90</b>
5.1	Notations . . . . .	91
5.2	Overview of Gridless Channel Routing . . . . .	93
5.3	Imposing the Bounding Constraints on Channel Routing . . . . .	95
5.3.1	Problem Formulation . . . . .	96
5.3.2	The Mapping Algorithms to Eliminate Critical Couplings . . . . .	97
5.3.3	Meeting Nonzero Bounds on Coupling Capacitances . . . . .	108
5.4	Imposing the Matching Constraints on Channel Routing . . . . .	111
5.4.1	Defining Symmetrical Channels . . . . .	111
5.4.2	Routing a Channel with Lateral Symmetry . . . . .	113
5.5	ART . . . . .	119
5.5.1	Program Input . . . . .	119
5.5.2	Program Operation . . . . .	120
5.6	Results of Channel Routing . . . . .	121
5.6.1	An Illustrative Example . . . . .	121
5.6.2	SC Filter Example . . . . .	122
5.6.3	A/D Converter Example . . . . .	125
5.7	Constraint-driven Area Routing . . . . .	126
5.8	Constraint-driven Placement . . . . .	127
5.9	Summary . . . . .	128
5.10	Appendix: Necessary condition for avoiding crossovers . . . . .	128

<b>6</b>	<b>Analytical Models for Capacitances</b>	<b>131</b>
6.1	The Approach . . . . .	132
6.2	Numerical Simulation . . . . .	134
6.3	Model Fitting . . . . .	138
6.3.1	Choosing a form for the polynomial . . . . .	138
6.3.2	Computation of coefficients of the polynomial . . . . .	139
6.3.3	Computation of highest powers of parameters . . . . .	140
6.4	Forms chosen for the configurations . . . . .	140
6.4.1	Single-Line Configuration . . . . .	141
6.4.2	Crossover Configuration . . . . .	142
6.4.3	Parallel Lines on the Same Layer . . . . .	144
6.4.4	Parallel Lines on Different Layers . . . . .	146
6.5	CAPMOD . . . . .	149
6.5.1	Program Input . . . . .	149
6.5.2	Program Operation . . . . .	150
6.6	Results . . . . .	153
6.6.1	Models for Single Line . . . . .	154
6.6.2	Models for Crossover . . . . .	155
6.6.3	Models for Parallel Lines on the Same Layer . . . . .	156
6.6.4	Models for Parallel Lines on Different Layers . . . . .	156
6.7	Summary . . . . .	159
<b>7</b>	<b>Conclusions and Future Directions</b>	<b>161</b>
7.1	Conclusions . . . . .	161
7.2	Future Directions . . . . .	163
	<b>Bibliography</b>	<b>165</b>
<b>A</b>	<b>User's Manual of PARCAR</b>	<b>174</b>
<b>B</b>	<b>User's Manual of ART</b>	<b>177</b>
<b>C</b>	<b>User's Manual of CAPMOD</b>	<b>181</b>



# List of Figures

1.1	choice of analog vs. digital processing . . . . .	1
1.2	Effects of parasitics on circuit performance. . . . .	4
1.3	The traditional approach for layout design . . . . .	6
1.4	The physical design problem . . . . .	7
1.5	Performance-constrained physical design . . . . .	7
2.1	Unity-Gain Bandwidth and Phase Margin of a circuit . . . . .	14
2.2	Slew rate of a circuit . . . . .	15
2.3	Single line over ground plane . . . . .	17
2.4	Multiconductor configuration . . . . .	18
2.5	Slicing-structure-based layout . . . . .	22
2.6	Nonslicing-structure-based layout . . . . .	23
2.7	A channel . . . . .	24
4.1	Parameters determining nominal precision. . . . .	48
4.2	Matching constraints. . . . .	54
4.3	Flexibility model. . . . .	58
4.4	Flow diagram of PARCAR. . . . .	64
4.5	Fifth-order low-pass SC filter. . . . .	66
4.6	Fully-differential CMOS opamp. . . . .	68
4.7	Closed-Loop Configuration. . . . .	69
4.8	Sensitivity with process variations. . . . .	73
4.9	Adjacent channels of a channel. . . . .	78
4.10	Coupling between horizontal segments. . . . .	82
4.11	Coupling between vertical segments. . . . .	85
4.12	Coupling between crossing segments. . . . .	86
5.1	A channel and its VC graph . . . . .	93
5.2	The constraint-mapping problem . . . . .	97
5.3	Dependence of crossover capacitance on pin positions . . . . .	99
5.4	Shielding achieved using one vertical segment for CASE 1 . . . . .	102
5.5	Shielding achieved using two vertical segments for CASE 1 . . . . .	103
5.6	Illustration of conditions in CASE 2 . . . . .	105
5.7	Constraining the horizontal segments of shield nets . . . . .	106

5.8	A symmetrical slicing structure and the associated channels . . . . .	112
5.9	(a) Nonoverlapping and (b) Overlapping matched pairs . . . . .	114
5.10	Almost-perfect mirror symmetry using I-connectors . . . . .	115
5.11	Almost-perfect mirror symmetry using I-connectors . . . . .	116
5.12	Nonuniform width of the special net . . . . .	117
5.13	Program flow of ART . . . . .	121
5.14	Channel Routed Without the Constraints . . . . .	122
5.15	Channel Routed with the Constraints . . . . .	123
5.16	fifth-order low-pass SC filter . . . . .	123
5.17	Comparison of results with and without the constraints for the SC filter . .	125
5.18	Algorithmic A/D converter . . . . .	125
5.19	Comparison of results with and without the constraints for the A/D converter	126
5.20	case 1 and case 2 . . . . .	130
6.1	Automatic generation of analytical models . . . . .	134
6.2	Mesh used for a two-dimensional configuration having three conductors . .	135
6.3	Neighboring points of $(i, j)$ in the mesh . . . . .	136
6.4	Flux components of single-line configuration . . . . .	142
6.5	Crossover configuration . . . . .	143
6.6	Components of mutual flux in a crossover . . . . .	143
6.7	Parallel lines on the same layer . . . . .	145
6.8	Parallel lines on different layers . . . . .	146
6.9	Components of mutual flux for nonoverlapping lines . . . . .	147
6.10	Components of mutual flux for overlapping lines . . . . .	147
6.11	Process parameters of a process with arbitrary levels of interconnects and dielectric interfaces . . . . .	149
6.12	Flow diagram of CAPMOD . . . . .	151
6.13	Flow diagram of CAPMOD . . . . .	152
6.14	Variation of coupling and correction capacitances with width $w_1$ in a crossover ( $w_2 = 1\mu m$ ) . . . . .	155
6.15	Variation of coupling and correction capacitances with width $w_2$ in a crossover ( $w_1 = 1\mu m$ ) . . . . .	156
6.16	Variation of coupling and correction capacitances with the separation $s$ for parallel lines on bottom layer ( $w_1 = w_2 = 1\mu m$ ) . . . . .	157
6.17	Variation of coupling and correction capacitances with the width $w_2$ for par- allel lines on bottom layer ( $s = w_1 = 1\mu m$ ) . . . . .	157
6.18	Variation of coupling and correction capacitances with $p$ for parallel lines in configuration (b) ( $e_l = e_r = 1\mu m$ ) . . . . .	158
6.19	Variation of coupling and correction capacitances with $e_l$ for parallel lines in configuration (b) ( $p = 2\mu m, e_r = 0\mu m$ ) . . . . .	159

# Chapter 1

## Introduction

CAD has been successfully applied to the design of digital VLSI systems. Digital CAD tools which have been developed and are in use today are logic synthesis tools, automatic placement and routing tools, extraction tools, timing analysis tools and many others. However, most of the electronic systems designed are not purely digital but in general are mixed analog/digital. For any application there is a choice between a digital implementation or an analog implementation of the system. Digital implementation is usually preferable for low to moderate speed and high-precision applications (e.g. digital audio), Analog implementation may be preferable for very high-speed applications (e.g. monolithic microwave integrated circuits for microwave applications) and low precision applications (e.g. analog VLSI for neural networks). For many applications an optimum combination of analog and digital processing may be the right way to go.

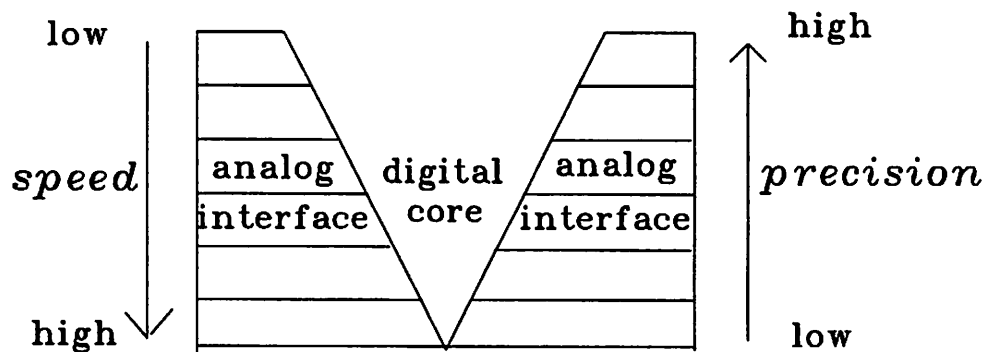


Figure 1.1: choice of analog vs. digital processing

Recently digital processing of signals is becoming popular for a wide variety of ap-

plications since the performance of a digital circuit is not very sensitive to device behavior, noise and parasitics. However, *real-world signals* are analog i.e. continuous in time and continuous in amplitude, whereas the input and output signals of a digital circuit are discrete in time and discrete in amplitude. Hence, there is need for analog circuits for interface purposes even when the host system is realized in the digital domain. Commonly desired analog interface operations for digital signal processing are A/D and D/A conversion, filtering and amplification.

The relative roles of analog and digital processing is pictorially illustrated in Fig. 1.1. For low-speed high-precision applications a minimal amount of analog interface can be used. For example for digital audio, an oversampled A/D converter can be used. There is no need for sharply limiting the bandwidth of the incoming signal according to the Nyquist criterion, and hence the input precision analog filter can be eliminated. The oversampled A/D operates at a clock rate much higher compared to the signal bandwidth, and the filtering is done in the digital domain. However this may not be feasible at higher frequencies (e.g. for video applications), as it would require the oversampled A/D to run at extremely high clock rates. Hence more analog interface may be required (an analog filter may be required before the A/D to limit the bandwidth of the signal). For very high-speed low precision applications it may be cost effective to do the entire processing in the analog domain.

About 80 percent of all printed circuit board designs, 50 percent of all full-custom integrated circuit design and 10 percent of all application-specific integrated circuits (ASICs) presently contain mixed analog and digital functions[28]. In future, ASIC designs would increasingly be mixed-signal in nature, as there is a strong trend for integrating both digital and analog functions on the same chip, to realize a high degree of *system integration*, and hence reduce system cost, and improve system performance <sup>1</sup>

At this time, while tools for digital design are well developed, tools for analog design are still insufficient to improve design productivity. Even if the analog portion occupies a small fraction of the total area of the system, it can become the design bottleneck when designed without the aid of CAD tools. Hence, today there is a strong need for extending the applicability of CAD tools to general mixed analog/digital systems.

---

<sup>1</sup>System cost is obviously reduced as less die and board area is occupied. System performance is also improved due to reduction of parasitics associated with interconnects running between chips.

## 1.1 Physical Design of Mixed Analog/Digital Systems

Traditional layout tools for digital VLSI try to minimize area of the chip subject to design rules. Examples of such tools are: (a) placement tools for automatic placement of blocks (b) routers for automatic realization of the interconnections between the blocks (c) compactors which transform an initial layout (not necessarily satisfying the design rules) to a layout (satisfying the design rules) while trying to minimize the chip area.

In mixed analog/digital systems there are many additional constraints involved in layout design due to analog circuits and the interaction between analog and digital circuits. These constraints stem from the fact that the performance of such systems is very sensitive to the stray effects in layout. Examples of performance functions of analog circuits are gain, bandwidth, phase margin, slew rate, gain, noise figure etc. Examples of stray effects would be parasitic capacitances, resistances, inductances, line-to-line parasitic coupling capacitances and mutual inductances.

Parasitics in a circuit can be broadly categorized into two groups: those associated with the devices, and those associated with the interconnects. Examples of a device parasitic would be source to bulk capacitance in a MOSFET. Device parasitics can be predicted after device dimensions are determined immediately after the electrical design phase.

The parasitics which are controlled during the various phases of automatic layout design such as placement, routing and compaction are the interconnect parasitics. The interconnect parasitics can be line resistances, capacitances, inductances or line-to-line capacitances and inductances (all of these parasitics can be considered as linear circuit elements). For monolithic circuits, these lumped approximations are valid for frequencies up to a few giga hertz (above which the transmission-line effect is evident).

Interconnect parasitics can play an important role in altering circuit behavior, if proper care is not exercised in layout design. For example, in switched-capacitor circuits, unwanted capacitive coupling between interconnects can destroy the ratio accuracy of precision capacitors. Noise signal fed from digital to analog lines can impair precision of analog circuits in a serious way, besides limiting their dynamic range. In an amplifier circuit, even a small capacitive coupling can degrade significantly the frequency response due to the Miller effect. Stray coupling which gives rise to positive feedback may lead to oscillations. Parasitic capacitances at high impedance nodes and parasitic resistances at low impedance nodes can critically affect circuit performance. Parasitic line inductances can be important

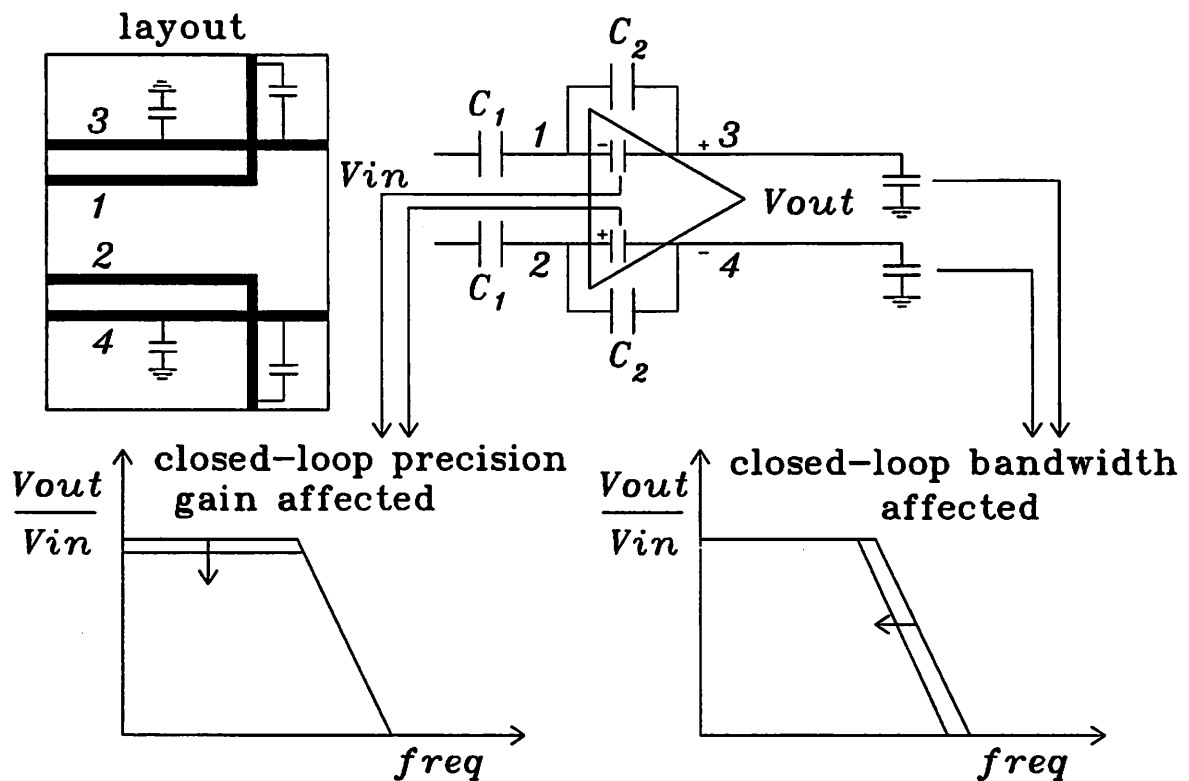


Figure 1.2: Effects of parasitics on circuit performance.

for low impedance nets and at high speeds of operation. Transient current in these nets give rise to inductive voltage drop in addition to resistive voltage drop. Often analog circuits employ fully differential architecture to achieve higher performance. This results in an additional need for symmetrical placement and routing for impedance matching and noise cancellation purposes.

The effects of interconnect parasitics have been illustrated in Fig. 1.2 for a fully differential circuit which uses an opamp and precision-ratioed capacitors  $C_1$  and  $C_2$  to realize a precision gain. The ratio accuracy of the capacitors can be of the order of 0.1%. The precision of the gain can be critical when the opamp is used in an A/D converter, since it affects the resolution of the conversion process. If the precision capacitances are of the order of  $0.1\text{pF} - 1\text{pF}$ , even a few  $f\text{F}$  of coupling capacitance between input and output terminals of the opamp can degrade the gain accuracy substantially. Coupling can arise due to crossing lines (each crossing contributes about  $2\text{fF}$  of coupling capacitance for a 3micron technology), or adjacent lines running parallel to each other (particularly significant for technologies employing aggressive design rules). Also, in the example shown,

if the interconnects associated with the output nets are long and the output nodes happen to be the high-impedance nodes of the opamp, the associated self capacitances will degrade the bandwidth of the circuit. This is pictorially illustrated in the layout portion of Fig. 1.2, where not only the interconnects have to be symmetrical with respect to net pairs (1,2), and (3,4), but also it may be necessary to bound the coupling and self capacitances of the interconnects shown.

Although device capacitances *to ground* are usually larger than the interconnect capacitances, the *line-to-line* coupling capacitances due to routing can be extremely critical. It should be noted that inter-node *device* capacitances can be present only between specific pairs of nodes, and the circuit designer is aware of them. On the other hand, *interconnect* capacitance can be present between any pair of nodes, and such a coupling which is not anticipated during electrical design when present in the layout, may sometimes be dangerous.

The layout of analog circuits has been traditionally designed manually. Analog designers spend a significant fraction of their design time in the physical design phase. The physical design usually proceeds in an iterative fashion as illustrated in Fig. 1.3. In each pass, after the layout is designed, all the layout parasitics are extracted and the circuit is simulated to check if performance specifications are met. But, normally the extracted list is huge, and if performance specifications are not met, no clue is obtained regarding what went wrong (the problem may not necessarily be due to a single parasitic, but can be due to the combined effect of several parasitics). Possible trouble spots in the layout are “guessed”, some changes made to the layout, and the process repeated. This is a highly inefficient approach, which can give rise to large number of time-consuming iterations. Hence, there is a need for a more disciplined approach towards physical design of analog circuits.

## 1.2 A Performance-Constrained Approach towards physical design

In this thesis, a general *performance-constrained* approach is proposed, the goal being to *drive* the layout tools by performance constraints, so that further layout iterations are either reduced in number or completely eliminated. The performance constraints during layout are considered to be the maximum changes allowed in performance functions

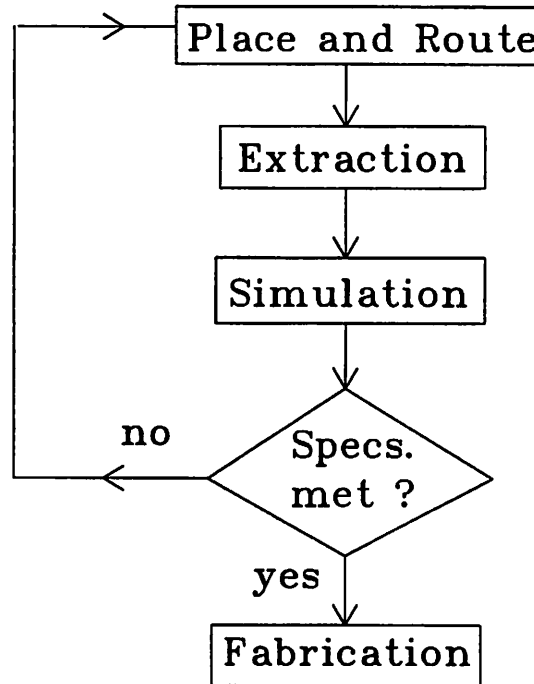


Figure 1.3: The traditional approach for layout design

because of layout parasitics to satisfy a set of specifications. The physical design problem is formulated as: *Given an electrical design of a circuit which meets all the performance specifications, carry out the physical design in such a way that the performance constraints are still met for the circuit with the additional parasitics introduced during layout design.* This is illustrated in Fig. 1.4 through the example of a switched-capacitor filter. The performance constraint in this case can be the maximum allowed change in ripple of the frequency response during physical design.

Since the high-level performance constraints of a circuit are too abstract for the layout tools to handle directly, a set of low-level constraints on the parasitics are generated from the performance constraints and used to drive the placement and routing. The parasitic constraints which we impose on the router are of two types: *bounding constraints* and *matching constraints*. A flow diagram of the approach is shown in Fig. 1.5.

Matching constraints are imposed based on impedance-matching and noise-cancellation considerations. This results in matching requirement between certain pairs of self capacitances(or inductances) and resistances, and also on certain pairs of coupling capacitances(or mutual inductances).



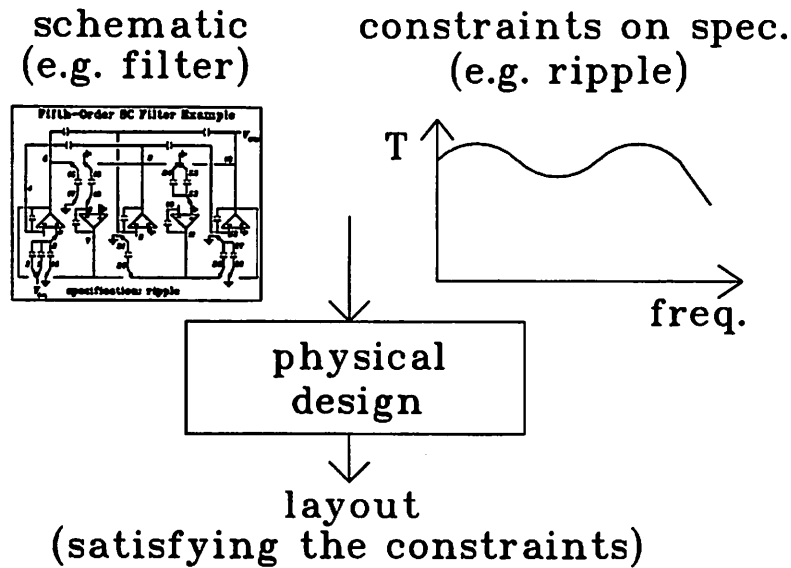


Figure 1.4: The physical design problem

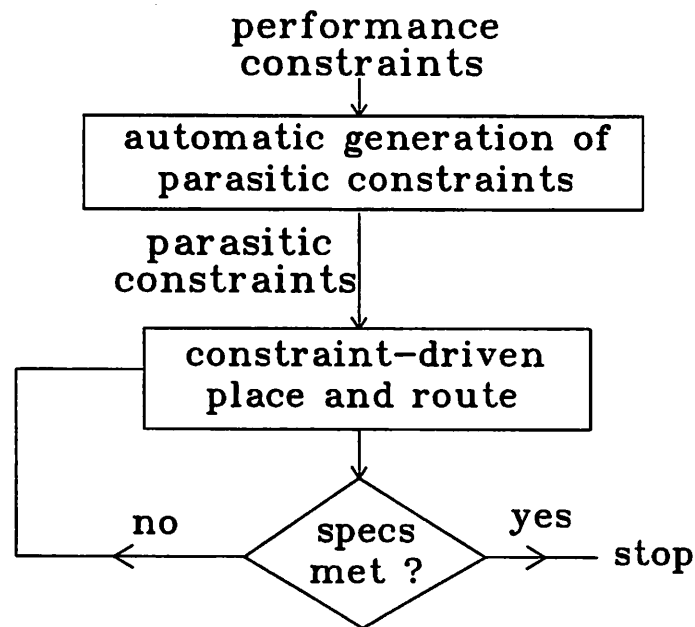


Figure 1.5: Performance-constrained physical design

Based on the performance-constraints, performance sensitivities (computed using a circuit simulator), and a-priori conservative estimates of maximum possible values of parasitics, the set of parasitics which can collectively cause performance degradation small compared to the maximum allowed, are ignored for generation of bounding constraints. The

remaining parasitics are called the critical parasitics.

The generation of bounding constraints on the critical parasitics is not straightforward due to two reasons. First, the performance functions are usually complex functions of the parasitics. Secondly, there are in general several possible combinations of the bounding constraints which may satisfy the performance constraints. The way each of these issues has been handled is discussed below.

Since during placement and routing, one wants to limit variations in performance functions to small values around their nominal values, linear approximations can be used to model dependence of performance on the parasitics, when the parasitics are within the bounds specified by the bounding constraints. Worst-case process variations are taken into account, and are used in worst-case modeling of the performance constraints.

In general, there are several parasitics affecting several performance functions. Hence, as mentioned earlier, there are several possible combinations of bounding constraints on parasitics which will satisfy the performance constraints. However, they may not all be equally easy for the router to meet. This is particularly the case when the parasitics have substantially different sensitivities. In that case, given one set of constraints it may be possible to tighten the constraints on more sensitive parasitics by small amounts and relax the constraints on less sensitive ones by large amounts to obtain another set of constraints easier to meet. The problem is formalized by introducing the notion of flexibility given to the router due to the bounding constraints, and then maximizing this flexibility subject to the performance constraints while generating the bounding constraints on the parasitics.

### **1.3 Organization of the thesis**

The thesis is organized as follows. Chapter 2 provides some background required for a better appreciation of this thesis. In this chapter basic concepts such as performance sensitivities and interconnect parasitics are defined and methods for their numerical computation described. It also has an overview of automatic physical design methodologies used today for digital VLSI. Chapter 3 has a survey of previous work involving analog design automation. From Chapter 4 onwards, the various phases of the performance constrained layout design proposed in this thesis are individually described, and relevant results presented. Chapter 4 embodies the automatic constraint generation procedure. This chapter presents algorithms for generating the parasitic constraints from the performance

constraints of the circuit. A prototype parasitic constraint generator PARCAR is described. Chapter 5 contains constraint-driven algorithms for layout design. Algorithms for imposing parasitic constraints on gridless channel routing are described. A constraint-driven channel router ART is presented. Interfacing of the constraint generator with the constraint-driven area router ROAD and constraint-driven placer PUPPY-analog are also briefly discussed. Chapter 6 describes CAPMOD, an analytical model generator for interconnect capacitances. It generates analytical models for interconnect capacitances based on numerical simulations and a partial knowledge of flux components associated with the configurations. These models are used in the constraint-driven layout design. Chapter 7 summarizes the contribution of this thesis and suggests possible future work.

## Chapter 2

# Thesis Background

This chapter provides some background required to appreciate the contents of the succeeding chapters which constitute the main contribution of this thesis. An overview of performance functions, performance sensitivities, interconnect parasitics and automatic layout design will be provided now.

### 2.1 Performance Functions

As discussed in Chapter 1, the performance-constrained approach followed by us involves computation of performance sensitivities. Performance sensitivities are sensitivities of performance functions of a circuit. Hence, first a brief discussion on performance functions of circuits is presented.

Performance functions of a circuit are the functions which describe its behavior. Depending on whether a performance function describes DC, AC or transient behavior, it can be a DC, AC or Transient performance function. Example of a performance function of an opamp circuit is unity-gain bandwidth, which is an AC performance function. It is defined as the frequency at which the magnitude of AC gain of the circuit goes to unity.

The performance function of a circuit will depend on several parameters of the circuit. The parameters of interest however depend on the particular design phase one is interested in. In the electrical optimization phase, the design parameters are the parameters of interest. The design parameters are parameters the circuit designer can control to achieve the required electrical performance. Examples of such parameters are widths and lengths of MOSFETS. While studying the effect of process variations for a given circuit, the process

parameters (e.g. threshold voltage and oxide thickness of a MOSFET) are the parameters of interest.

During placement and routing, one introduces interconnect parasitics which are in addition to the device parasitics already existing. While considering the interconnect parasitics, one has to initially consider all possible parasitics which may possibly exist in the layout. For example, in case of capacitances, one has to consider capacitances to ground from all the nodes, and coupling capacitances between all possible pairs of nets.<sup>1</sup>

## 2.2 Performance Sensitivities

In this thesis, the performance functions and the parasitics are respectively denoted by the letters  $W$  and  $p$ . Let a circuit have  $N_w$  performance functions denoted by  $W_1, W_2, \dots, W_{N_w}$ , and  $N_p$  parasitics denoted by  $p_1, p_2, \dots, p_{N_p}$ . The sensitivity of a performance function  $W_i$  with respect to a parasitic  $p_j$  is denoted by  $S_{ij}$  and is defined as the partial derivative of  $W_i$  with respect to  $p_j$ .

$$S_{ij} = \partial W_i / \partial p_j \quad (2.1)$$

Hence there will be  $N_w * N_p$  sensitivities to deal with. One way to compute sensitivity is by using the perturbation method. In this method for computing the sensitivity of a performance with respect to a parameter, the parameter is perturbed and the performance reevaluated using circuit simulation. The difference between the two performance functions divided by the perturbation in the parameter gives the sensitivity. However, this method is very time consuming since a resimulation of the circuit is required for each parameter of interest. This method can also be very error-prone particularly for transient simulations, since the circuit simulator output has some inherent error, and while taking difference between two close numbers, the percentage error gets amplified.

Sensitivities can be computed much more efficiently and accurately, compared to perturbation method by using direct or adjoint techniques of sensitivity computation. Sensitivity analysis (except possibly the adjoint transient sensitivity analysis) is supported in standard circuit simulators such as SPICE3[38], SPICE2[11] and TISPICE[40].

---

<sup>1</sup>However, as mentioned earlier, only critical parasitics will be automatically detected and finally considered.

A brief description of the direct method of sensitivity computation in SPICE is now provided. For more detailed description refer to [38][11]. For transient analysis, SPICE solves a set of differential equations using numerical integration methods. The differential equations are of the following form:

$$\partial x/\partial t = f(x,p) \quad (2.2)$$

where  $x$  is the vector of circuit variables (node voltages and branch currents).  $p$  is the vector of parameters of interest. The sensitivity  $\partial x/\partial p$  of  $x$  with respect to  $p$  is a matrix. The  $(i,j)^{th}$  element of the sensitivity matrix represents the sensitivity of the  $i^{th}$  circuit variable  $x_i$  with respect to the  $j^{th}$  parameter  $p_j$ . Differentiating the circuit equations with respect to  $p$ , one gets:

$$\partial(\partial x/\partial t)/\partial p = \partial f(x,p)/\partial p \quad (2.3)$$

$$(2.4)$$

Interchanging the order of differentiation with respect to  $p$  and  $t$ , one gets:

$$\partial(\partial x/\partial p)/\partial t = \partial f/\partial p + (\partial f/\partial x)(\partial x/\partial p) \quad (2.5)$$

Hence, a differential equation in terms of the sensitivity is obtained. At each time point, the circuit variables and the sensitivities of the circuit variables with respect to the specified parameters are computed using numerical integration methods (for solving differential equations). The Jaconian matrix computed at each time point for solving the circuit differential equations can be reused for solving the sensitivity differential equations, considerably reducing the overhead due to sensitivity computations. If the sensitivity analysis with respect to a specific parameter is requested, when the simulation is over, the time domain waveforms and the transient sensitivity waveforms for that parameter are simultaneously available from the simulator. The DC sensitivity is only a special case of transient sensitivity at zero time.

For AC analysis, SPICE linearizes the circuit at each frequency point and computes a set of small-signal parameters which are functions of the DC quiescent voltages and currents. The small signal parameters are used in the circuit equations. The AC circuit equations can thus be written as follows:

$$J(p, a(p))A(p) = R \quad (2.6)$$

where  $A$  is the complex vector of AC circuit variables (node voltages and branch currents),  $a$  is the vector of DC quiescent voltages and currents.  $p$  is the vector of parameters of interest and  $R$  is the vector of independent sources.  $J$  is the Jacobian matrix. Differentiating the equation above with respect to  $p$  and moving terms to the other side of the equation one gets :

$$J(\partial A/\partial p) = [(\partial J/\partial p) + (\partial J/\partial a)(\partial a/\partial p)]A \quad (2.7)$$

The AC sensitivity  $\partial A/\partial p$  is computed by solving this set of equations. As is obvious, the DC sensitivities of the quiescent voltages and currents are also considered for computing the AC sensitivities.

It is to be noted that the direct method simultaneously computes sensitivities of all the circuit variables with respect to a given parameter. In SPICE3, the additional CPU time required for each parameter is about 8% for transient sensitivity, 4% for ac sensitivity and 1% for DC sensitivity. When the number of parameters is large, the direct method of sensitivity computation can be expensive. The adjoint technique[42] can be preferred in that case. In the adjoint method, sensitivities of one circuit variable is computed at a time, but with respect to all the parameters of interest. When the number of parameters is more than the number of circuit variables whose sensitivities have to be computed, then the adjoint method is more efficient than the direct method for DC and AC sensitivity computations. For transient sensitivity computation adjoint method involves substantial amount of additional computations and is extremely difficult to implement.

Circuit simulators such as SPICE compute sensitivities of circuit variables (node voltages and branch currents). Hence, intermediate calculations have to be performed to compute the sensitivities of the performance functions [39]. This process is illustrated for three typical performance functions of analog circuits, the unity-gain bandwidth, the phase margin and the slew rate.

As shown in fig. 2.1, the unity-gain bandwidth  $f_u$  is defined by the equation

$$[g(f)]_{f=f_u} = 1 \quad (2.8)$$

where  $g(f)$  represents the ac gain of the circuit as a function of frequency  $f$ . Taking its derivative with respect to a parameter  $p$ , one gets

$$[(\partial g/\partial f)(\partial f/\partial p) + (\partial g/\partial p)]_{f=f_u} = 0 \quad (2.9)$$

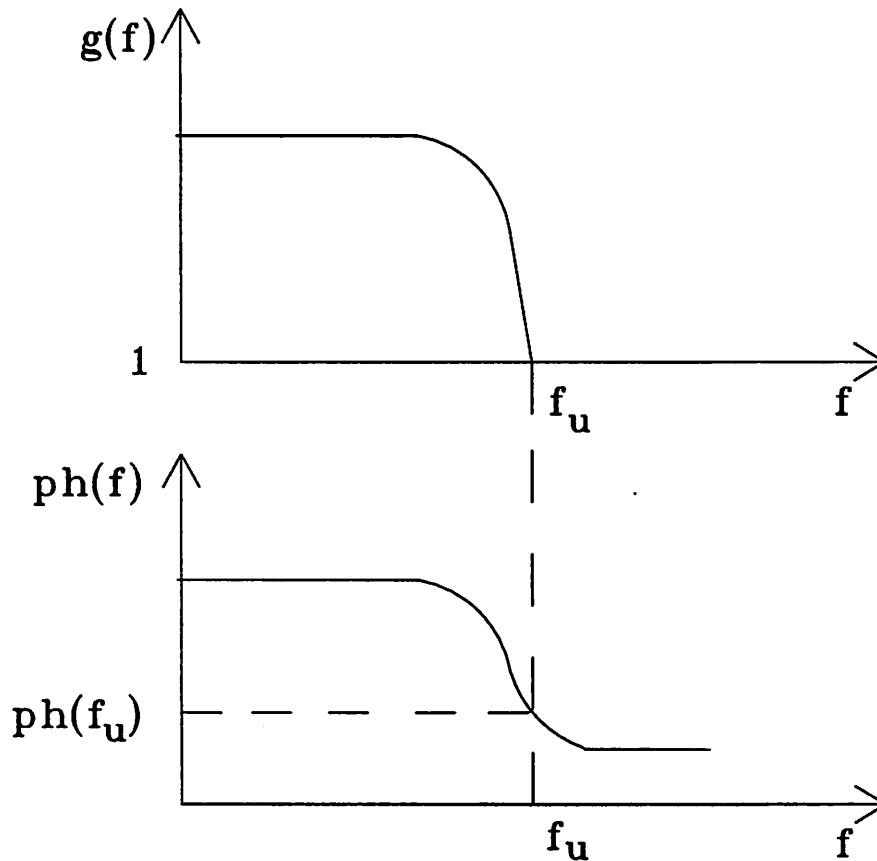


Figure 2.1: Unity-Gain Bandwidth and Phase Margin of a circuit

$$\partial f_u / \partial p = -[(\partial g / \partial p) / (\partial g / \partial f)]_{f=f_u} \quad (2.10)$$

Hence the sensitivity of the unity-gain bandwidth is computed by computing  $\partial g / \partial p$  and  $\partial g / \partial f$  at unity-gain frequency.  $\partial g / \partial p$  can be computed by the circuit simulator.  $\partial g / \partial f$  is the slope of the gain response.

The phase margin of a circuit is defined by the equation

$$pm = [ph(f)]_{f=f_u} + 180 \quad (2.11)$$

where  $ph(f)$  represents the ac phase response of the circuit as a function of frequency  $f$  as shown in Fig. 2.1. Taking its derivative with respect to a parameter  $p$ , one gets

$$[\partial pm / \partial p]_{f=f_u} = ([\partial ph / \partial f]_{f=f_u})(\partial f_u / \partial p) + [\partial ph / \partial p]_{f=f_u} \quad (2.12)$$

$$(2.13)$$



Substituting the expression for  $\partial f_u/\partial p$  from Eq 2.10

$$[\partial pm/\partial p]_{f=f_u} = -[(\partial ph/\partial f)(\partial g/\partial p)/(\partial g/\partial f)]_{f=f_u} + [\partial ph/\partial p]_{f=f_u} \quad (2.14)$$

$$(2.15)$$

$\partial g/\partial p$  and  $\partial ph/\partial p$  can be computed by the circuit simulator.  $\partial ph/\partial f$  is the slope of the phase response curve at the unity-gain-frequency.

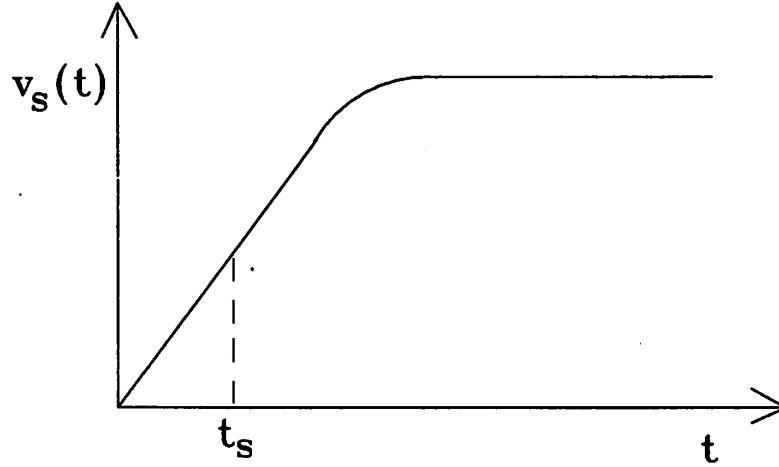


Figure 2.2: Slew rate of a circuit

The slew rate of a circuit is defined to be the slope of the rising portion of the step response  $v_s(t)$  as shown in Fig. 2.2. If the slope is measured at time  $t_s$  in the rising portion of the response, then slew rate  $sr$  is given by

$$sr = v_s(t_s)/t_s \quad (2.16)$$

Taking the derivative of the above equation with respect to a parameter  $p$  we get

$$\partial sr/\partial p = ([\partial v_s/\partial p]_{t=t_s})/t_s \quad (2.17)$$

where  $\partial v_s/\partial p$  is computed by the circuit simulator for each time point during the transient sensitivity analysis.

### 2.3 Interconnect Parasitics

As mentioned in Chapter 1, the parasitics which are controlled during placement and routing are the interconnect parasitics. The parasitics can be in the form of resistances,

capacitances and inductances. A brief overview is provided describing the techniques of their computation.

### 2.3.1 Resistances

Any interconnect has a series resistance associated with it. The resistance  $R$  of an interconnect is given by

$$R = \rho * l / (w * t) \quad (2.18)$$

where  $\rho$  is the resistivity of the material,  $l$  is its length,  $w$  is its width and  $t$  its thickness. This expression is valid if the current distribution inside the conductor is assumed to be uniform. This is usually true for low frequencies. At high frequencies, such as microwave frequencies, current distribution becomes nonuniform inside the conductor due to the *skin effect*. As a result the resistance changes with frequency.

Resistance of an interconnect is also affected by the presence of discontinuities like bends, vias etc. The numerical computation of the resistance of general structures using finite-element technique is reported in [43].

### 2.3.2 Capacitances

Capacitance computation is more complicated than resistance computation, because of the capacitive coupling between interconnects.

As the widths of interconnection lines are scaled down, the thickness is either kept constant or scaled by a much smaller factor to limit resistance. This causes the fringing effects to dominate the interconnect capacitances, causing gross error in estimation, if the parallel-plate approximation is used to compute line-to-ground and crossover capacitances. Also, the increasing ratio of thickness to separation between lines has the effect of increasing substantially the coupling capacitance between two adjacent parallel lines, which is again a complicated function of various layout parameters.

Theoretical analysis for accurate modeling of capacitance is of limited practical use, since it cannot be applied to the complicated configurations encountered in the layout. Numerical methods based on finite-difference, finite-element or integral-equation techniques can be used to extract the desired electrical parameters [56]-[63] of any structure. However, these methods usually involve very high computational cost, and are not meant for repeated use in large layouts.

Several analytical expressions have been suggested for the line-to-ground capacitance of a single-line configuration [44]-[50]. A comparison of the various models is provided in [55].

Two of the empirically obtained models are included here. In these models,  $c$  is the capacitance per-unit-length (F/m).  $\epsilon$  is the permittivity of the medium.  $w$  and  $t$  are respectively the width and the thickness of the conductor, and  $h$  is the height of the conductor above the ground plane as shown in Fig. 2.3.

Sakurai and Tamaru Model[49]:

$$c = \epsilon[1.15 * (w/h) + 2.8 * (t/h)^{0.222}] \quad (2.19)$$

This model has an accuracy of about 6% for  $0.3 < w/h < 30$  and  $0.3 < t/h < 30$ .

Meijs and Fokkema model[50]:

$$c = \epsilon[(w/h) + 0.77 + 1.06(t/h)^{0.25} + 1.06(t/h)^{0.5}] \quad (2.20)$$

This model has an accuracy of about 2% for  $w/h > 1$  and  $0.1 < t/h < 4$ , and of about 6% for  $w/h > 0.3, t/h < 10$ .

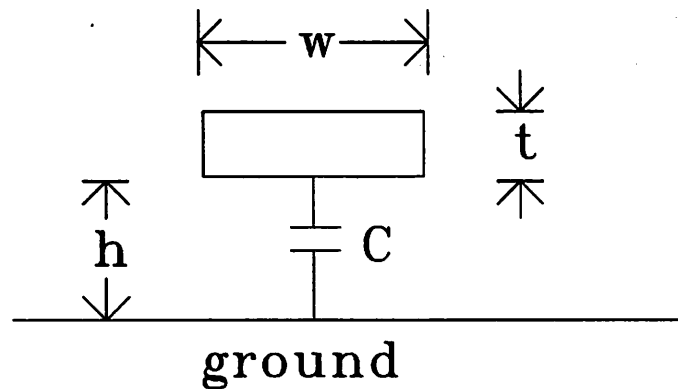


Figure 2.3: Single line over ground plane

Besides a line-to-ground capacitance for each interconnect, there also exists a coupling capacitance between any two lines. This coupling is strong when lines run parallel to each other or cross each other. Analytical expression for line-to-line coupling capacitance of adjacent parallel lines is reported[51]. Expressions for coupling capacitances tend to be very complicated.

Let us consider a general multiconductor configuration consisting of  $N$  interconnects (an example of a multiconductor configuration is shown in Fig. 2.4 which has  $N$  parallel lines). Let  $C_{ii}$  denote the capacitance to ground of the  $i^{\text{th}}$  interconnect, and  $C_{ij}$  denote the coupling capacitance between the  $i^{\text{th}}$  and the  $j^{\text{th}}$  interconnect. Let the voltage on the  $i^{\text{th}}$  conductor be  $V_i$ . Then the charge  $Q_i$  on the  $i^{\text{th}}$  conductor is :

$$Q_i = C_{ii}V_i + \sum_{j=1}^N C_{ij}(V_i - V_j) \quad (2.21)$$

$$\text{or } Q_i = (C_{ii} + \sum_{j=1}^N C_{ij})V_i - \sum_{j=1}^N C_{ij}V_j \quad (2.22)$$

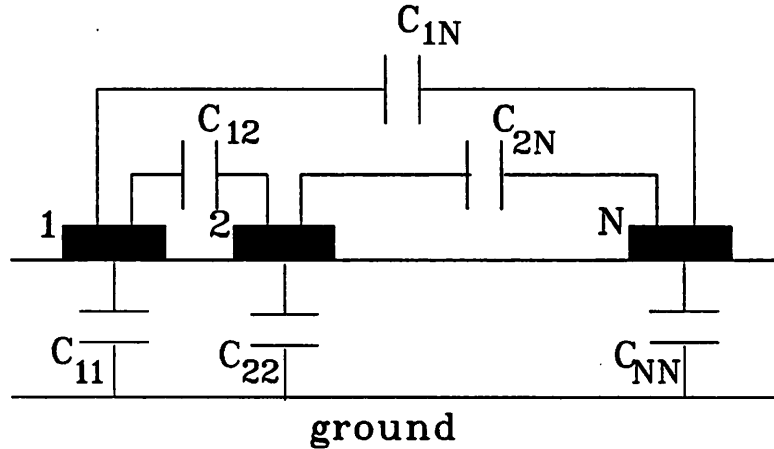


Figure 2.4: Multiconductor configuration

Hence the charge voltage relationship can be compactly written as

$$Q = [C]V \quad (2.23)$$

where

$$C = \begin{bmatrix} C_{11} + \sum_{j=1}^N C_{1j} & -C_{12} & \dots & -C_{1N} \\ -C_{21} & C_{22} + \sum_{j=1}^N C_{2j} & \dots & -C_{2N} \\ \cdot & \cdot & \dots & \cdot \\ \cdot & \cdot & \dots & \cdot \\ -C_{N1} & -C_{N2} & \dots & C_{NN} + \sum_{j=1}^N C_{Nj} \end{bmatrix}$$

$$Q = [Q_1, Q_2, \dots, Q_N]^T \quad (2.24)$$

$$V = [V_1, V_2, \dots, V_N]^T \quad (2.25)$$

$C$  is called the capacitance matrix of the multiconductor configuration. The  $(i, i)^{th}$  diagonal element of the matrix is  $C_{ii} + \sum_{j=1}^N C_{ij}$ . This is called the total capacitance of the  $i^{th}$  conductor, and is the sum of its capacitance to ground and its coupling capacitances with respect to all other conductors. The  $(i, j)^{th}$  element is the negative of the coupling capacitance between the  $i^{th}$  and the  $j^{th}$  conductor.

Numerical techniques such as the finite-difference method described later in this thesis, compute this capacitance matrix by successively setting the voltage of each conductor to  $1v$  and of all the other conductors to zero, and computing the charges on the various conductors. From eq. 2.22, it is obvious that if  $V_i = 1$ , and  $V_j = 0, j \neq i$ , then  $Q_i$  directly gives the total capacitance of the  $i^{th}$  conductor, and  $-Q_j, j \neq i$  directly gives  $C_{ij}$ . Subtracting the sum of these coupling capacitances from the total capacitance gives the capacitance to ground of the  $i^{th}$  conductor.

### 2.3.3 Inductances

For a multiconductor configuration with  $N$  interconnects, let  $L_i$  denote the self inductance of the  $i^{th}$  interconnect and  $M_{ij}$  denote the mutual inductance between the  $i^{th}$  and the  $j^{th}$  interconnect. Let  $I_i$  and  $\Phi_i$  denote respectively the current and magnetic flux associated with the  $i^{th}$  interconnect. Magnetic flux is associated with a closed loop. In a configuration such as the one shown in 2.4, ground closes the loop for each interconnect. The flux  $\Phi_i$  is then given by

$$\Phi_i = L_i I_i + \sum_{j=1}^N M_{ij} I_j \quad (2.26)$$

$$(2.27)$$

Hence the flux-current relationship can be compactly written as

$$\Phi = [L]I \quad (2.28)$$

where

$$\Phi = [\Phi_1, \Phi_2, \dots, \Phi_N]^T \quad (2.29)$$

$$I = [I_1, I_2, \dots, I_N]^T \quad (2.30)$$

$$L = \begin{bmatrix} L_1 & M_{12} & \cdot & \cdot & M_{1N} \\ M_{21} & L_2 & \cdot & \cdot & M_{2N} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ M_{N1} & M_{N2} & \cdot & \cdot & L_{NN} \end{bmatrix}$$

$L$  is called the inductance matrix of the multiconductor configuration. Let us consider a two-dimensional configuration, (which has no variation in its dimensions in one direction). An example of such a configuration is the one shown in Fig. 2.4. Let the medium in which the conductors are present be nonmagnetic, a valid assumption for integrated circuits. Let  $C_0$  be the capacitance matrix for the multiconductor configuration with all the dielectric replaced by air. There exists a relationship between the inductance matrix and the capacitance matrix  $C_0$ [64].

$$[L] = \mu_0 \epsilon_0 [C_0]^{-1} \quad (2.31)$$

where  $\mu_0$  and  $\epsilon_0$  are respectively the permeability and permittivity of air.

However, the assumption made while deriving the relation above is that the current in each conductor is on the surface which is a good approximation for high frequencies. Inductance computation taking into account the current distribution inside the conductors is reported in [67]. Numerical techniques can be applied to compute accurately the inductances of three-dimensional structures[65] A detailed survey of numerical computation of interconnection parasitics is provided in [66].

## 2.4 Automatic Physical Design

Automatic placement and routing form the backbone of computer-aided physical design. Only a brief review of traditional automatic physical design which has been used

for digital VLSI will be provided in this chapter. For more detailed surveys refer to [68][69].

### 2.4.1 Placement

Given a set of components (which can be devices or higher-level macros), the placement problem is traditionally formulated as assigning the positions of the components in such a manner that the final layout occupies minimum area. Recent trend is for considering also the timing constraints of a digital circuit during placement. However the final layout is available only after the interconnections are realized during the routing phase. Hence during placement only an approximation of the quality of the final layout is used to obtain a good placement. Many existing placement tools use estimates of total net length and area to evaluate the quality of placement.

The length of a net represents a signal delay along the net. Different nets can be assigned different weights based on their criticality. Computation of the shortest length of multi-terminal net is a Steiner-tree problem and is NP hard<sup>2</sup>. But, the semiperimeter of the shortest bounding box enclosing the terminals can be used as an estimate of the net-length.

The layout area is the area occupied by the components and the routing area needed for interconnections. The routing area estimate for a given placement can be made by performing a global or loose routing. Global Routing is a phase in which the nets are assigned to regions (such as channels) whereas their exact positions are not determined. When the nets are assigned to a channel, the density of the channel (maximum number of nets crossing the channel at any position) can be determined. This gives a good estimate of the height of the channel and hence of the routing area. However, performing global routing along with placement is expensive. Another approach to obtain an estimate of the area is by expanding each edge of a component by an amount proportional to the number of pins on that edge[76].

The placement of blocks can be categorized into two broad classes (a) slicing structures (b) non-slicing structures.

In a slicing-structure-based placement, the positions of blocks can be visualized to be obtained by successively slicing the layout. An example of a slicing structure is shown in Fig. 2.5 (a). In this layout there are three slices (1, 2 and 3). A slicing structure-based layout can be obtained by using the min-cut method proposed in [72] for printed circuit

---

<sup>2</sup>There is no polynomial-time algorithm known to solve this problem

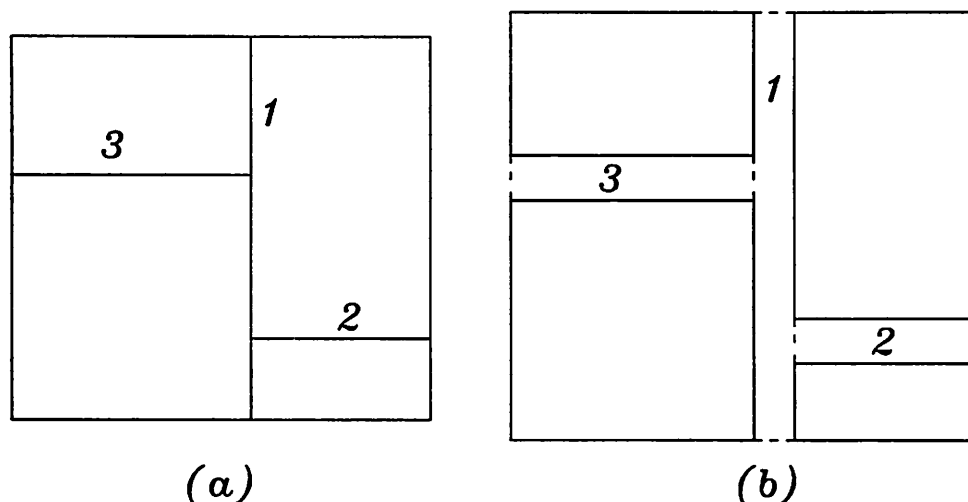


Figure 2.5: Slicing-structure-based layout

boards, gate arrays and standard cells and in [73] for macro cells. This method is based on recursive application of a bi-partitioning algorithm such as the Kernighan-Lin Partitioning algorithm[74]. Simulated Annealing[75] can also be used to optimize the placement for a slicing structure.

One of the advantages of having a slicing structure is that the layout can be completely routed using channel routers, which have enjoyed popularity for VLSI. For example, in Fig. 2.5(b), the three channels (1, 2 and 3) corresponding to the three slices in Fig. 2.5 (a) are shown. Moreover, given the relative positions of the modules in the form of a slicing structure, there are algorithms to find optimal orientation[70] and shape[71] of each block.

A nonslicing-structure-based layout (example shown in Fig. 2.6 is more general and may provide some area saving over the slicing-structure-based layout. However, it may not be possible to route completely the layout using a channel router. For example, in Fig. 2.6, channels 1,2,3,4 and 5 can be routed in the this order, but region 6 would have fixed pins on all four sides and hence has to be routed using an area router. The nonslicing-structure-based layout can be formed using several methods[68], such as the greedy clustering approach, force-directed methods, eigen-value methods and simulated annealing. The Timber Wolf system[76] using simulated annealing produces placement of excellent quality, although it is CPU intensive.



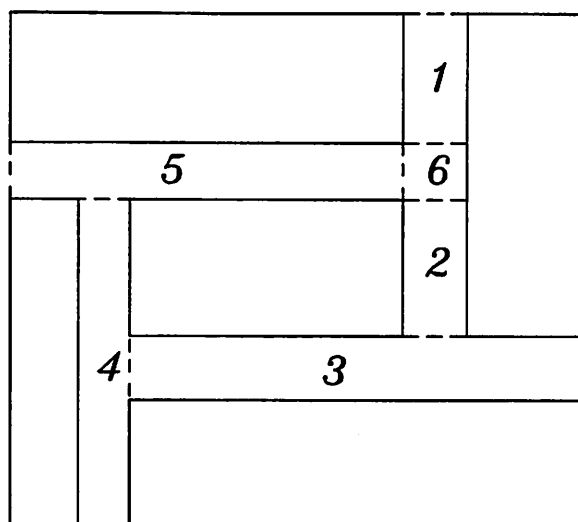


Figure 2.6: Nonslicing-structure-based layout

### 2.4.2 Routing

Given a set of components and their pin positions, routing is the problem of realizing the interconnections in minimum area. There are two broad categories of routing algorithms: (a) Channel Routing (b) Area Routing.

In channel routing, first proposed in [77], the routing problem is decomposed into smaller problems of routing rectangular regions called channels. The routing region can be completely broken into channels for a slicing structure based layout as mentioned before. A channel (an example shown in Fig. 2.7) has fixed pins on the top and bottom edges and floating pins on the left and right edges. When an edge has a floating pin of a net, it implies that the net crosses that edge, and the exact crossing position will be determined after routing. Many of the existing channel routers can handle irregular top and bottom edges enhancing the applicability of such routers.

After the placement of blocks a global routing is performed which determines the paths of the nets in terms of the channels they have to go through. As a result one obtains the floating pins on left and right edges of the channel. Global routing is performed in such a way that the density of each channel does not exceed a bound (constraint for gate arrays), or the overall size of the layout is minimized (for standard cell and macro cell layout styles).

After global routing, the individual channels have to be routed in a particular order[78] to complete the detailed routing. As an example, in Figure 2.5, channels 2 and

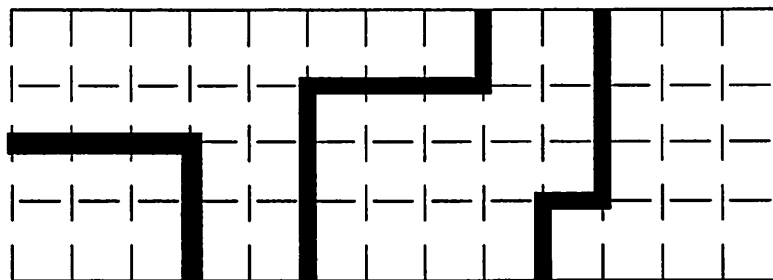


Figure 2.7: A channel

3 have to be routed before channel 1. In grid-based channel routing, the routing region is subdivided into rows or tracks and columns as shown in Fig. 2.7. The pins are at the positions of columns. Typically, one assumes that the vertical and horizontal segments are routed in different layers when two layers are used for interconnects. The goal is to complete the routing to minimize the number of tracks. Several channel-routing algorithms have been proposed in the literature[79][80] [81][82].

Area routing is a more general form of routing. In this form of routing, the routing region can be any irregular shape and can have obstacles and pins at arbitrary positions. Many area routers are based on some modified version of Lee-Moore maze algorithm[83]. The routing region is represented in the form of a grid. The router routes one net at a time along the grid. Given two pins to be connected, in the Lee-Moore algorithm a wave is propagated from the source pin till it hits the destination pin. It is followed by a back-trace to obtain the minimum-cost path of the net. The routed net is treated as an obstacle for routing the next net. A good net ordering is important for successful routing. Area routing may often involve ripping-up some or all nets at any stage and rerouting. The router MIGHTY[84] is based on an incremental routing technique. It modifies the connections already implemented to allow bad interconnections to find a better implementation.

Some area routers use the line search algorithm[85]. This is a gridless algorithm in which horizontal and vertical lines are expanded from both the pins (to be connected). If the lines originating from the two pins do not intersect because of obstacles, lines are expanded again from the points they hit the obstacles. This process is repeated till they intersect. A path is obtained using a back-trace along the lines.

When two layers of interconnects are used, channel routing has several advantages over area routing. Since a channel router routes all the nets at the same time, one achieves

routing of excellent quality, compared to area routing strategies that route one net at a time, and also two pins at a time. Since channel routing breaks the routing into smaller problems, it is usually much faster compared to the general area-routing algorithms. Channel routing also allows for changes in placement of blocks with relative ease during detailed routing. Hence it is not necessary to know the routing area in advance. Area routing on the other hand may be useful while routing very irregular geometries when channels can not be easily defined.

Many of the new technologies employ more than two layers of interconnects to realize denser integration. Two-layer channel algorithms can be generalized to handle multi layers of interconnects[86]. But, for sufficiently large number of layers there may not be any need for assigning channels for routing, as the area over the cells may be sufficient to accomodate the interconnections. For such technologies area routing is obviously preferable.

### 2.4.3 Spacing/Compaction

Spacing is the problem of transforming an initial layout (not necessarily satisfying design rules) to a layout satisfying the design rules and possibly other constraints such as symmetry while attempting to minimize area. A spacer is used along with a symbolic layout system [88] to obtain a mask-level layout. This way, prior to spacing, the layout can be completely designed at the symbolic level substantially reducing the complexity of the data to be maintained. Moreover, symbolic layout makes the layout technology-independent. A spacer is also used to obtain a compact layout from another layout which is not densely packed (this is why spacing is also called compaction).

Several spacers such as FLOSS[87], STICKS[89], CABBAGE[90], SPARCS[23] have been reported in the literature. Most of the existing compacters are based on the one-dimensional approach, i.e. they decompose the two-dimensional compaction problem into two one-dimensional problems. In this approach, first the y-dimensions of all the blocks are fixed, and the x-dimensions are varied to minimize the x-dimension of the layout. Then the x-dimensions are fixed and the y-dimensions varied to minimize y-dimension of the layout. This procedure is repeated till convergence of the total area under some criterion is achieved. In CABBAGE, compaction in each dimension is performed using a constraint graph and a longest-path technique. The objects in the layout are represented by nodes in the constraint graph and the design-rule constraints existing between the design-rule constraints existing

between the objects are mapped to weights of arcs between corresponding nodes in the graph. The length of the longest directed path in the graph determines the minimum dimension of the layout in the direction of compaction. The nodes on that path (also called the critical path) represent the objects whose positions are determined directly from the weights on the arcs to minimize the total length in the dimension of compaction. There is some flexibility in deciding the positions of other objects which have to be placed within some range again decided by the weights of the arcs.

There have been attempts to compact both x and y dimensions of the layout at the same time. This may cause more area saving as more degrees of freedom are exploited. However, it can be CPU intensive. Two-dimensional compaction using Boolean variables and functions is reported in [91] and using zone-refining is reported in ZORRO[92].

## Chapter 3

# Previous Work in Analog Design Automation

The synthesis process for analog circuits can be subdivided into two main tasks: (a) circuit synthesis and (b) layout synthesis. Circuit synthesis systems generate an electrical circuit which satisfies a set of high-level performance specifications. Layout synthesis systems obtain a layout considering the analog-specific issues.

### 3.1 Previous work in Circuit Synthesis

The circuit synthesis problem for analog circuits consists of three stages (a) topology selection (b) initial circuit design (c) circuit optimization. During topology selection, the interconnections of building blocks (which are e.g. transistors for an opamp design) is determined to come up with a circuit topology having the potential to meet a set of performance specifications. After a particular topology is selected, the initial circuit design problem is to assign a set of appropriate values to the design parameters of the circuit (e.g. widths and lengths of the transistors for a CMOS opamp design) to obtain a feasible design, i.e. a design meeting the performance specifications. However, the initial design is only “a” design satisfying the specifications. One may like to have the best design, where “best” describes the design having minimum value of some cost function (the cost function can be, e.g., the total area occupied by the circuit). The circuit optimization problem is to obtain a design which minimizes a prescribed cost function subject to a set of constraints on performance functions of the circuit (the specifications).

The general circuit synthesis problem in case of analog circuits is much more difficult than for digital circuits. This is because of the fact there are no fundamental algorithms known to synthesize a topology of the analog circuit from its specifications. In case of digital circuits any logic can be realized using only NAND or NOR gates. However no such fundamental building block is available for analog circuits. New topologies are discovered only through human intuition. Hence, till today the circuit synthesis work has focussed on choosing a suitable topology from a number of fixed alternatives and then optimizing its performance.

The circuit synthesis approaches described so far can be broadly categorized into two categories : (a) knowledge-based approaches: which use rules and heuristics to synthesize an analog circuit (b) analytical approaches: which model the performance functions of some known analog circuits using a set of analytical expressions and synthesize a circuit to satisfy the performance specifications. However, it is often difficult to model complex performance functions of analog circuits accurately using analytical expressions. Hence there is a need for a CAD tool which can optimize the performance functions of any analog circuit based on circuit simulation. The tools which perform this task are called circuit optimizers. First, a survey of work on circuit-synthesis systems is provided. Then a brief survey of previous work on general-purpose circuit optimizers will be provided.

### 3.1.1 Synthesis systems

AID2, which is an automated analog IC design system [14] was one of the first systems to be reported for analog design. It uses module compilers to produce fixed and programmable (parametrized) cells, which are then placed and routed in a standard cell layout system. Automatic synthesis of switched-capacitor filters[14] and of successive approximation A/D and D/A converters[15] were reported. Examples of parametrized cells are capacitor arrays and resistor strings, which are programmable with respect to the number of bits decoded by each in an A/D converter. Interconnections between these cells are defined by a program which translates high-level specifications into predefined circuit description.

However, the work reported above has not presented any systematic way of obtaining circuit parameters from the specifications. A survey of knowledge-based synthesis systems which have addressed this problem is now provided. Then, analytical approaches

for solving this problem are discussed.

Synthesis by knowledge-based hierarchical decomposition of circuits was reported in [1]. This work presented the PROSAIC system which synthesizes CMOS operational amplifiers. This system stores analog circuits as a hierarchy of circuits. Synthesis is performed by first decomposing the design into independent sub-circuits until the level of complexity can be handled by the designer or the system. The global specifications of the circuit are then used to impose specifications for the individual subcircuits using the knowledge base. The selection of appropriate topologies for the subcircuits are then made from the data base to satisfy the performance specifications and port requirements (for proper connection to other subcircuits). Electrical constraints arising from interconnections between the subcircuits are considered and necessary alterations made to the subcircuits. Overall circuit performance is evaluated and if it does not satisfy the specifications, the sequence of steps described is repeated.

After PROSAIC, an expert system BLADES using subcircuit experts was reported in [3]. The system works based on circuit partitioning, and use of subcircuits as standard building blocks. The system has a general design manager which talks to a number of subcircuit experts, which are specialists in particular subcircuit designs. For a given set of specifications, each subcircuit expert will provide the design manager with appropriate subcircuits. The design manager schedules and coordinates activities among the subcircuit experts, and critiques the overall system design through various system-level consultants such as simulators like SPICE. If the specifications are not met, the designs of subcircuits are changed by the expert system. Five subcircuit designers are used, one each for current sources, differential amplifiers, voltage references, voltage sources and level shifters. The knowledge base for each subcircuit is divided into different categories. Each category may then have different realizations. For example, the current source knowledge base is divided into four categories : (a) NPN, (b) PNP, (c) NMOS and (d) PMOS. The NPN current source knowledge base in turn has 15 types of designs ranging from "Widlar" current source to a bandgap reference current source.

Knowledge-based approach with a planning mechanism was then reported in [4], which presents the OASIS system synthesizing CMOS operational amplifiers from performance specifications and process parameters. In this framework analog circuit topologies are represented by a hierarchy of functional blocks. Each functional block has a fixed number of alternate realizations. A planning mechanism is introduced to translate performance

specifications from one level of hierarchy to the next lower level in the hierarchy. Plans are stored with fixed topologies, and executed when the topology is used. Circuit equations are stored for each circuit topology. Heuristics are used that force the algebraic constraints to be satisfied at each step in the translation process. Rules fire at the end of each plan step to correct errors and change the flow of the plan. This mechanism can be illustrated by the gain partitioning of a two-stage amplifier circuit. A plan mechanism is first executed to do this partitioning considering the performance specifications such as phase margin. At the end of the plan step it may be discovered that the overall gain is not achievable. A rule then fires to see if either stage is in cascoded configuration. If not, one stage is cascoded to increase the gain. The process of translation of specifications down the hierarchy continues till it reaches the bottom of the hierarchy and determines the behavior of the devices. At this point the sizes of all the devices are determined.

The three systems reported above rely on a hierarchical style of synthesizing circuits based on decomposition of the circuit specifications into those of components using knowledge embedded in the system. The following work also uses successive decomposition of specifications but uses macromodelling for more efficient modelling.

In [7], analog compilation based on successive decompositions of high level specifications and physical assembly requirements into those of lower level subblocks and devices is reported, the driving applications being High Voltage ASICs. The decomposition is driven by template knowledge. A solution space is constructed to permit only useful decompositions, using heuristics and analog design knowledge. Macromodelling is used to simulate and verify the decomposition at any stage. This prevents frequent depth-first searches in the solution space. Finally, a mixture of behavioral and structural compilation is used, for easier satisfaction of analog design and layout requirements. A compiler An\_com has been developed to compile analog functional blocks at the complexity of operational amplifiers.

Like the approaches described above, the following work uses knowledge base for synthesis, but chooses one among a set of fixed circuit topologies, rather than relying on hierarchical decomposition to obtain the circuit.

The IDAC system reported in [6] can synthesize a wide range of analog CMOS circuits such as transconductance amplifiers, operational amplifiers, low-noise BIMOS amplifiers, voltage and current references, quartz oscillators, comparators, and oversampled A/D converters including their digital decimation filters. The user specifies the technology, the desired building-block specifications and the design options. For performing a worst-



case design the user is also required to specify minimum and maximum values of electrical parameters of devices, the matching of components, layout rules needed to calculate source and drain capacitances and extreme temperature and supply voltage values. Finally the user selects one or more schematics in the library. The program sizes the devices in the schematic by choosing three types of knowledge, i.e. knowledge specific to the schematic, general circuit knowledge (e.g. sizing cascode devices) and knowledge common for a family circuits (e.g. stabilizing an amplifier). After the sizing phase, the program uses an analyzer to evaluate the correctness of design. If the analyzer finds that some specifications are not met, then a new set of target specifications is defined. If all specifications are met, the program checks if the used device models are valid for the particular design. It tries to detect potential leakage, high-frequency effects, short-channel effects etc. Comparison of measured performance and performance predicted by the program has been provided for several synthesized circuits and a good match has been reported. Use of this program enabled the designers to perform electrical design by exploring many degrees of freedom in a short time, e.g. a 13-bit A/D converter could be designed only in 4 hours (elapsed time).

As it is difficult to formalize the way human designers design circuits, the knowledge-based synthesis systems described above attempt to embody as much human expertise as possible, but may fail in particular situations. Analytical approach to synthesis is an alternate approach as mentioned earlier. Two systems based on analytical approach for opamps and switch-cap filters are now described.

In [5], OPASYN, a synthesizer for CMOS operational amplifiers is reported. This system has a topology database, an optimization module, an interface to the circuit simulator SPICE and a parameter update module. The topology database has a fixed number of circuit topologies and also analytical models of performance functions for each topology. Based on the application, the system chooses a circuit topology from the topology database. For each circuit topology there is a declaration of independent design parameters and a reasonable range of values for these parameters. A coarse grid sampling of the entire parameter space is used, and from the more promising sampling points found in this survey, a steepest-gradient descent algorithm searches for the optimal solution in its neighborhood. The cost function used in this algorithmic optimization is defined as a sum of number of penalty terms where each penalty reflects the violation in a particular performance target. An exponential term is used to model this violation. As a result, when the actual performance violates the target by a certain margin, the positive contribution to the cost function

is much larger than the negative contribution which will result if the performance satisfies the target by an equal margin. Thus, the cost function tries to prevent certain performance functions from being violated by large margins which can otherwise result because some other performance functions satisfy their targets by equivalently large margins. The system often returns several locally optimal solutions. The SPICE simulator interface automatically simulates the optimized circuit and determines more accurately the performance parameters. The SPICE simulation results are also used by the parameter-update module to improve the analytical models (by finetuning some parameters in the model) in the data base. For DC convergence, before simulation, initial node voltages are obtained from a simulation of a slightly modified circuit topology which has much better dc convergence properties.

A switched-capacitor filter compiler has been reported in [2]. It is capable of designing Butterworth, Chebyshev, Cauer, and Thompson filter classes of low pass, high pass, band pass, and band elimination types. The program automatically calculates the poles and zeros of a normalized version of the requested filter type from the high-level specifications. The compiler selects the required biquad building blocks from a collection of parametrized cells and the layout generated with automatic interconnections.

The analytical approaches are much more formal than knowledge-based approaches, but their main drawback is that it is often difficult to model the performance of any general analog circuit analytically. Moreover, even if analytical models are used they may not be accurate.

### 3.1.2 General-purpose Circuit Optimizers

General-purpose circuit optimizers optimize the performance of an analog circuit, given an electrical design and a simulator to evaluate its performance. In gradient-based optimization, the gradients of the performance functions are evaluated either using finite-difference or using numerical sensitivity capability of the simulator. Several circuit optimizers have been reported in the literature, some of which are described below.

The AOP circuit optimizer[8] is based on the circuit simulator ASTAP[9] with adjoint sensitivity analysis capability. The multiple objective functions and constraints in optimization are combined into one cost function, with the constraints formulated as penalty terms. This program was used to minimize the memory cycle of a read-only memory cell.

APLSTAP[10] is an interactive optimizer using the simulator ASTAP. It uses interactive linear programming and presents to the user linear prediction of circuit performance at the solution of each LP step.

DELIGHT.SPICE[11] and ECSTASY[12] are circuit optimizers based circuit simulators SPICE2 and SPICE3 respectively. They use nonlinear gradient-based optimization methods for circuit optimization. ECSTASY also has a controlled random search algorithm. These optimizers can handle ordinary and functional constraints (constraints which depend not only on the design parameters but also on an independent parameter such as time, frequency and temperature). A good value and a bad value can be specified for each performance function. The optimizer first tries to meet the bad values for all the performance functions, and then attempts to meet the good values. In DELIGHT.SPICE, the optimization problem has to be described using the RATTLE language. ECSTASY has a form-based graphical user interface and is much more user friendly.

The optimizers are very powerful tools if a simulator with sensitivity analysis capability is available for the circuit of interest. Often for mixed analog/digital circuits suitable simulators are not available. Moreover, even if a simulator is available, when transient performance functions (such as slew rate) are involved, repeated evaluation of such performance functions makes the optimization very time consuming. Often a viable solution to synthesis problem is to first obtain a circuit close to its optimum design by a synthesis system (which uses knowledge-based or analytical approach), and then obtain a more exact optimization using the circuit optimizers.

## 3.2 Previous Work in Layout Synthesis

As in case for digital VLSI, the layout synthesis problem of analog circuits is composed of several phases such as placement, routing and compaction. Work on layout design automation reported in the literature falls into two categories: (a) layout systems capable of generating the layouts for specific types of circuits such as opamps, filters (usually these systems form part of complete synthesis systems for those specific types of circuits), (b) more general-purpose layout synthesis systems capable of synthesizing the layout of a broad class of analog circuits from the schematics (c) tools such as a placer, a router or a compactor (the purpose of these tools is to find use as either design aids for analog designers, or form part of some synthesis systems).

### 3.2.1 Special-purpose Layout Systems

First, an overview of some layout synthesis programs reported for specific types of circuits are described.

ADORE, a program for layout synthesis of SC filters is reported in [16]. The floor-plan consists of an array of capacitors located in between an array of operational amplifiers and an array of switches. The opamps are chosen from a standard cell library. The switches and capacitor arrays are automatically generated to suit the high-level specifications on the filter. The relative placement of blocks is optimized using simulated annealing. The channels between the arrays are routed using a channel router. A postrouting step is used to shield the crossovers between the sensitive and the the large-swing nodes. This is done assuming there are three layers available for routing. Horizontal and vertical segments of the channels are routed in the first and third layers. The shield plates are formed at the sites of unwanted crossovers in the second layer. Then an area router is used to connect the shield plates to ground. This program is able to synthesize layout comparable in quality to manual design. However, since standard cells are used for opamps, the program has cannot adapt to scaling of technology.

The layout generator of the opamp synthesis program OPASYN [19] has parametrized leaf cell generators for single mosfets, transistor pairs and capacitors. For each of the circuit topologies in the data base, a circuit-dependent slicing tree is used for the floorplanning of blocks based on the knowledge obtained from designers. The device sizes are determined during circuit optimization, as mentioned earlier. For the placement of blocks, the program does a bottom-up traversal of the slicing tree, considering all possible aspect ratios of the leaf cells. This is followed by a top-bottom traversal in which the user-specified constraint on aspect ratio and area are mapped into the dimensions of the devices. Routing is performed using a switch-box router. Wires are assigned to categories with higher priorities given to nets having additional constraints such as matching requirements. Layout spacing is carried out using a compactor. The good thing about this program is that it optimizes aspect ratios of devices to minimize area. However, since it uses a fixed topology for the placement of layout components, optimization in area is limited. Since the routing is carried out by essentially a digital router, the analog constraints are not handled completely.

CADICS, a technology-independent synthesis tool for cyclic A/D converters was reported in [36]. This tool is capable of synthesizing A/D converters in a broad range of

sampling rate, resolution and silicon area without using any standard cells. Depending on the specifications, a particular netlist module generator is first chosen. Individual blocks are then optimized to meet the specifications. The netlist generated by the circuit synthesis is structured to illustrate how the A/D converter is partitioned. The layout synthesis portion uses a stacked transistor layout style involving leaf-cell generation, placement and routing. For a given structured netlist, a top-down and then a bottom-up hierarchical layout procedure is performed. Since both analog and digital blocks are involved in the design, special consideration is given to isolation of analog and digital blocks during placement. Routing is carried out by essentially a digital router. hence, making it difficult to handle analog constraints. Work is in progress to interface the constraint-driven analog router ART developed at U.C. Berkeley to the system.

### 3.2.2 General-purpose Layout Systems

“General-purpose” here means the capability of designing layouts of a broad class of circuits whose technologies are supported by a layout system.

Early work on layout synthesis was carried out at AT&T. A standard-cell placement strategy to eliminate coupling between *zero-swing-sensitive* nets and *large-swing-insensitive* nets has been described in [13]. In this design style the layouts of the cells are designed in such a way that pins associated with sensitive nodes (e.g. virtual ground of an opamp) are on the top edges of the cells, pins associated with the large-swing/insensitive nodes (e.g. output of an opamp) are on the bottom edges of the cells. The cells are oriented upside down in alternate rows of the standard-cell layout. As a result, the sensitive and the large-swing/insensitive nets are routed in alternate channels. The insensitive nets are further classified into many classes. Bias and supplies to tubs containing switches are routed first along the edge of the cell, followed by analog large-swing nets, analog ground and finally digital control signals. As a result, the analog power rails are shielded from analog-large swing nets and analog large-swing nets shielded from digital control signals. The sensitive nets are always restricted inside the respective sensitive channels. The cells of a subcircuit (such as a filter) are always restricted to lie within some predefined bounds of the subcircuit to restrict the sensitive nets. An order of magnitude increase in design productivity and fully functional “first silicon” for various chips such as line scan circuit, digital subscriber loop, fiber optic modem, transversal filter, optical data link receiver and

equalizer/echo canceller have been reported. The maximum complexity reported is for a chip with 11600 transistors. This approach takes care of some of the problems involved in routing analog circuits. Some constraints like symmetry constraints are not handled. Moreover, this design style imposes some restrictions on the design as cells have to be designed in a particular way. Also, it is possible to have more than one group of sensitive wires which have critical coupling constraints with respect to each other. For example, analog circuits often contain *finite-swing-sensitive* nets, such as the input of a sample and hold, inputs of a differential amplifier, etc. which have to be isolated from other sensitive nets as well as from other large-swing nets. Hence putting the sensitive nets in alternate channels does not completely solve the problem.

After the AT&T work, several other approaches have been reported which provide more flexibility to the designer during layout design. To illustrate this, overview of two systems: ILAC and ANAGRAM, are now provided.

ILAC[18], is a layout tool which automatically generates the geometrical layout of analog CMOS leaf cells from netlist information and user-specified constraints on cell bounds and input/output locations. The circuit is partitioned into functional blocks which can be basic devices such as transistors, resistors, capacitors or structures such as current mirrors, differential pairs, cascodes etc. Each block can have a number of possible physical realizations (variants). For example, a MOS transistor can have the following variants: rectangular, snake, interdigit, circular, concentric, S-shape, and waffle. The layouts of the blocks are generated by the block generators which can handle analog constraints such as matching constraints (e.g. for matched centroidal bipolar transistors) and symmetry constraints (e.g. for differential pairs). Once suitable layouts of the blocks are generated, an initial floorplan is created based on a slicing structure and some simulated-annealing-based preoptimization. This is followed by simulated-annealing-based placement optimization including global routing. Blocks can be defined to be in clusters, matched groups or symmetry groups. During simulated annealing moves, blocks in a symmetry group are kept symmetrical with respect to some axis, blocks in a matched group have the same variant and orientation, and those in a cluster group are physically kept together, by having them in one subtree of the slicing structure tree. After placement optimization, detailed routing is performed using a scan-line based incremental channel router. This router routes net-by-net in a given priority order. It routes supply nets first in a single layer. Then it routes the sensitive nets trying to minimize their lengths and the parasitics. After this, insensitive

nets are routed followed by noisy nets. During routing, a cost function involving unwanted coupling, switching of layers and area increase is minimized. The placement and routing are performed at the symbolic level. A compactor is then used to obtain the physical layout.

ANAGRAM, a tool for full-custom layout of analog cells using macro-cell design style is reported in [20]. KOAN/ANAGRAM II which is an improved version of ANAGRAM is reported in [37]. KOAN is a simulated-annealing-based placer. The placement algorithm allows nonslicing structure and has constrained move sets to handle symmetry and device matching. Suitable device merging and abutment are performed during placement to reduce layout area and parasitics. This is achieved by adding a negative contribution to the cost function which is proportional to the weighted sum of area and perimeter saved by a merging. Examples of abutment/merging are MOS gate abutment, MOS diffusion sharing, BJT guard-ring merging, MOS metal abutment routing, MOS substrate contact sharing and capacitor contact sharing. Wells are generated after the placement of devices, by expanding the geometries of the well-bound devices. This facilitates dynamic device merging during placement. After the placement, routing is performed using the line-expansion type area router ANAGRAM II. It uses a tile-plane representation of the layout, in which all devices, wiring space and wires are represented in a single-tile-plane data structure. The router sees the internal details of the devices and this results in easier over-the-device routing. The line-expansion router always maintains a set of partial paths sorted by the cost. The router proceeds by choosing the path with minimum cost and expanding it in several possible directions resulting in more partial paths. To avoid unwanted coupling, the cost function of the router has a term proportional to the crosstalk between sensitive and noisy signals. The router can perform symmetric routing of differential nets in the presence of asymmetrical obstacles. Layouts competitive with manual layout were reported.

In both ILAC and ANAGRAM, a cost function which has contributions from parasitics is minimized during routing. However, weights can be assigned in several possible ways to the parasitics to include them in the cost function, and the quality of resulting layout will depend on these weights. Minimization of a cost function with no reference to performance functions of the circuit does not guarantee satisfaction of performance constraints.

The works reported below use expert systems to perform analog layout automation.

In [17], a methodology for automatic layout design based on partitioning of the analog circuit into functional blocks consisting of library cells is presented. The system allows the user to provide an incomplete specification of the basic cells, and using an algo-

rithm similar to that used in pattern recognition, finds the best match between the specified descriptors and the cells in the library. An example would be an opamp for which slew rate and output voltage range have been specified. If the program finds several opamps in the library which satisfy these specifications it will choose the one with the best combination of other unspecified specifications such as offset voltage and cell area. The system adds and modifies descriptors used for selection process from previous knowledge. The system also has a set of primitive cells. Example of a primitive cell would be a unit resistor used to form resistor strings, and the set of descriptors would be: accuracy, voltage range, absolute resistance value, current density, maximum operating frequency, parasitic capacitance, leakage current, voltage coefficient, temperature coefficient, and interconnect resistance. After the cells are selected, a macrocell layout is pursued. Placement of macrocells is performed interactively, and the I/O peripherals are added automatically. Maximum values of interconnection parasitics (resistances and capacitances) can be assigned for the autorouter. A 16-bit angular shaft encoder chip has been designed using this system in less than 2 weeks.

SALIM[21], a layout generation tool for analog ICs, uses a combination of algorithmic and rule-based approach. The layout generation is a bottom-up process, in which initially a grouping is performed by identifying elementary functions called structures such as differential pairs, current mirrors. Each structure has several variants and the associated shape functions to represent their deformability. These shape functions are used to optimize aspect ratios of the cells. Placement is obtained using a force-directed algorithm, followed by global routing and detailed channel routing. Rules are used to handle analog-specific constraints during placement and routing. A new environment is reported which can merge procedural languages and inference rules.

SLAM, an analog module generator reported in [22] proceeds through three steps (a) circuit primitive recognition, (b) critical net analysis, (c) placement and routing. In the first step, in addition to basic circuit nodes, four critical analog circuit nodes are specially recognized : high-impedance nodes, diode-connected nodes, source-coupled nodes and current-mirror nodes. It also stores six kernel circuit primitives (ranging from a single transistor to cascode current mirror). Based on the information obtained from recognized circuit nodes and primitives, the topological configuration of the circuit is recognized. In the second step, based on the information about the circuit topology, the circuit nodes are classified into four classes (a) power, (b) sensitive, (c) noisy and inactive nodes. This classification imposes constraints on lengths of the nets, and separation between nets during



placement and routing. Matching constraints derived from topology information are also handled during placement and routing. Fabricated chips designed using this system has been reported.

LADIES, an automatic layout system for analog LSI's is reported in [27]. An initial layout is obtained based on a clustering algorithm for placement. During clustering, virtual nets with positive weights are created between elements to be placed together and with negative weights created between elements to be placed far apart. Gridless river routing is performed net-by-net after global routing. Throughholes are created afterwards if routing cannot be completed in one layer. After an initial layout is obtained, it is improved based on (a) compaction knowledge which tries to reduce area and (b) constraint knowledge which tries to reduce violation of analog-specific constraints. Layout constraints involving relative positions of components and/or wires can either be specified by the user or are generated by a rule-based preprocessing program. Layout of an D/A converter which is 8% larger than manual layout has been reported.

Expert system based approaches described above use rules to handle analog-specific issues. However, these rules are usually very case-specific and may not be able to encompass all the constraints involved in layout design. It is difficult to satisfy the performance constraints of an analog circuit by relying on a set of rules.

Some recent layout systems have reported new contributions in the placement arena. Three such systems are reported.

A sensitivity-based automatic layout compiler has been reported in [30], at the same time as the first report of this doctoral work[31]. It models the layout interconnect parasitics as equivalent circuits of RC networks. The sensitivity of the performance with respect to layout interconnect parasitics is analyzed. Partitions for placement are then based on circuit structure, results of the sensitivity analysis and the boundary conditions. Routing is then performed with the MIGHTY[84] program. However, in this approach, the constraints imposed are qualitative in nature and no quantification of performance degradation is done. Moreover routing is performed using a digital router which is unable to handle the analog constraints

In the two approaches described below, the physical placement of components is derived from their relative positions in the schematic.

ALE, a layout generating and editing system for analog LSIs has been reported in [32]. The layout generator generates the layout preserving the relative positions of device

cells and wires. Hence physical constraints in the circuit diagram are reflected in the layout. A two-layer maze-running algorithm is used for routing. The layout editor has interactive tools such as cell merging tool and the cell form modification tool.

In [34] a layout generator for bipolar analog LSI has been reported. A placement algorithm is reported which determines cell positions based on the relative positions of devices in the input circuit diagram. A device position graph is defined which has information about the relative positions of devices in the circuit schematic. The placement is obtained by grouping the vertices of this graph and laying the devices associated with the groups in different rows. A fine-grid and variable-width maze router is used then for routing.

However, in the two systems reported above, constraining the relative positions of layout components may affect area optimization. Moreover, parasitic constraints are not handled well in these approaches.

### 3.2.3 Layout Tools

Several layout tools have been reported tools such as routers and compactors, which can be used as components in analog layout systems.

An algorithm for two-layer gridless channel-routing of mixed analog/digital nets is reported in [28]. The channel-routing problem is represented by a vertical-constraint graph. The algorithm controls the net crossovers between mutually sensitive nets by appropriate arrangement of horizontal segments which is reflected as directed edges in the graph. Coupling between horizontal segments in the channel is also controlled to keep the critical capacitances within some specified bounds, by controlling the separations between the segments (weights of the appropriate edges in the graph). The algorithm also attempts to perform single-layer routing for the nets which have such restrictions (like the supply nets or nets which have to be routed in the low resistance layer). This is achieved by attempting to avoid crossovers with other nets. However, this work does not address the issue of how the bounds on parasitics can be obtained. Moreover, use is not made of shield nets to control coupling and symmetrical routing is not handled.

A routing methodology is reported in [33] which consists of two parts: (a) symbolic routing and (b) detailed routing. During symbolic routing, relative positions of wires are determined, whereas during detailed routing, actual positions are determined from technology information. Nets are classified into various categories (high-sensitivity, low-sensitivity,

noisy, power supply, bias). Electrical constraints on parasitics associated with the nets are classified according to functional classes of nets. During symbolic routing, interconnect parasitics are symbolically evaluated using discrete values depending on the tracks of the individual nets. The most important electrical constraint is considered to reduce the search space for determining the path of each net. The second most important constraint is used to further limit the search space and so on. The net ordering is determined by an expert system. To match certain parasitics additional stubs are used. However, this approach cannot keep the routing parasitics under any specified bounds, as constraints are handled at the symbolic level.

Global and detailed channel routing to minimize crosstalk noise between signal lines is reported in [35]. Signals nets are classified into four groups: (a) high impedance analog (most sensitive), (b) low impedance analog (less sensitive), (c) analog/digital (e.g. digital control signals affecting the analog cells) (d) other digital nets (most insensitive). Nets are given priority based on the category they belong to. Digital and analog nets are routed in different areas. Double-width lines are used to avoid voltage drops due to parasitic resistances. A postprocess puts shield lines in specific positions. This approach like the previous one cannot keep routing parasitics under specified bounds. Moreover, matching of parasitics is not handled.

RoAD[29] is an area router for analog layouts which uses a modified version of Lee's algorithm. It can perform symmetrical routing for differential architectures. Control can be exercised over parasitic resistances, capacitances and coupling capacitances by a set of user specified weights. These weights automatically modify the cost function of the router and hence the quality of routing. For example, if a large weight is assigned to the resistance of a net, wave expansion (used in modified version of Lee's algorithm) will be quick in the low-resistivity layer. Hence paths in the low-resistance layers will get preference over paths in high-resistivity layers. If a large weight is given to the coupling between two nets, then a cost-function hill is created around one of the nets. When the other net is routed the associated wave will be repelled away from the hill and finds a path away from the other net even though it may be longer than the geometrically shortest path. Although in this approach, parasitics can be controlled by increasing or decreasing weights, there is no way to directly control performance degradation of the circuits directly.

Compaction considering symmetry constraints has been reported in SPARCS[23], CAMELEON[24] and in [25]. A special type of constraint called "active constraint" is used

in [23] based on a graph-theoretic perturbation method to force the relative spacing between one pair of components to be the same as the relative spacing between the other pair. A “class constraint” based on Integer Linear Programming is used in [24], and a combination of graph-based and linear programming method is used in [25]. However, these approaches do not handle other constraints like parasitic constraints, which may impose constraints on lengths and relative separation of specific nets.

In [26], compaction has been reported, which can handle various layout specifications important for analog circuits such as placing devices closely or at some distance, preventing certain pairs of wires from crossing each other, and maintaining specified net structures. Compaction is done by using longest directed path method, and the various specifications are mapped to the constraint graph. A clean-up function is used after compaction to reduce detoured wire patterns and eliminate unnecessary vias. This compaction considers more analog constraints than the other works mentioned, but again does not attempt to meet the performance constraints

### **3.3 Common Limitations of Previous Approaches in Analog Layout Design Automation**

A fundamental limitation of all the previous approaches towards layout design is that no quantification of performance degradation is made during layout design. As noted before, the performance of an analog circuit is sensitive to layout parasitics which gives rise to large number of time consuming iterations. With this motivation a generalized *performance-constrained* approach towards physical design of analog circuits is proposed in this thesis, the goal being to reduce the need for expensive layout iterations.

Moreover, in the previous approaches the issue of modelling parasitics for layout design is not handled. Accurate modelling of parasitics often requires numerical simulation (involving finite-difference, finite-element and integral-equation techniques) as discussed in the next chapter. However, these numerical techniques are too expensive for repeated use in layout. In the approach suggested in this thesis analytical models for parasitics are generated automatically by a model generator by performing numerical simulations. These analytical models are then used during constraint-driven layout design.

## Chapter 4

# Parasitic Constraint Generation

This chapter and the next two constitute the main contribution of this dissertation. As mentioned in the introductory chapter, the performance-constrained approach proposed in this dissertation consists of two distinct phases: (a) parasitic constraint generation, (b) constraint-driven layout design. This chapter deals with the first phase, and presents algorithms for automatic generation of parasitic constraints from high-level performance constraints.

### 4.1 Notation

The notation used in this chapter is as follows.

$\alpha$ : ( $0 < \alpha \ll 1$ ) used to set the limit for performance degradation caused by the noncritical parasitics

$a_j, b_j, c_j$ : coefficients of the polynomial modeling the layout flexibility  $f_j$

$b$ :  $N_{cp} \times 1$  column vector whose  $j$ th element is  $-b_j$

$C(n_i, n_j)$ : interconnect capacitance between  $n_i$  and  $n_j$

$C(n_i)$ : interconnect capacitance from  $n_i$  to ground

$f$ : total layout flexibility associated with all the parasitics

$f_j$ : layout flexibility associated with parasitic  $p_j$

$M$ : set of matched pairs of nodes in the circuit

$N_{cp}$ : number of critical parasitics

$N_{mc}$ : number of matching constraints

$N_i^+$ : number of parasitics noncritical for positive constraint on  $W_i$

$N_i^-$ : number of parasitics noncritical for negative constraint on  $W_i$

$N_p$ : number of parasitics

$N_q$ : number of process parameters

$N_w$ : number of performance functions

$p$ :  $N_{cp} \times 1$  column vector having  $p_{j\_bound}$  as  $j$ th element

$p_0$ : fictitious parasitic which replaces a matched pair of parasitics

$p_{j\_bound}$ : bound on parasitic  $p_j$  to satisfy performance constraints

$p_{j\_max}$ : conservative estimate of maximum value of  $p_j$

$p_{j\_min}$ : conservative estimate of minimum value of  $p_j$

$p_{thresh}$ : threshold values of the parasitics (below which any parasitic can be ignored)

$q$ : process parameter

$\Delta q_{j\_max}$ : maximum variation of  $q_j$  around its nominal

$Q$ :  $N_{cp} \times N_{cp}$  matrix used in the cost function of quadratic programming

$R$ : the set of parasitics noncritical for all the performance functions

$R_i$ : the set of parasitics noncritical for performance function  $W_i$

$r$ : additional degradation in precision due to parasitics allowed as a fraction of nominal precision

$r_{max}$ : maximum fractional process variation of a parasitic

$r_{mis}$ : maximum fractional mismatch between matched parasitics

$S_0$ : sensitivity of a performance  $W$  with respect to fictitious parasitic  $p_0$

$S_{ij}$ : sensitivity of performance  $W_i$  with respect to parasitic  $p_j$

$S_j$ : sensitivity of performance  $W$  with respect to parasitic  $p_j$

$S_{ij}^-$ :  $-S_{ij}$  if  $S_{ij} \leq 0$ , otherwise zero

$S_{ij}^+$ :  $S_{ij}$  if  $S_{ij} \geq 0$ , otherwise zero

$S_i^-$ :  $1 \times N_{cp}$  row vector whose  $j$ th element is  $S_{ij}^-$

$S_i^+$ :  $1 \times N_{cp}$  row vector whose  $j$ th element is  $S_{ij}^+$

$S_{lo}$ : lower bound on sensitivity  $S_{ij}$  due to process variations

$S_{up}$ : upper bound on sensitivity  $S_{ij}$  due to process variations

$W$ : performance function

$W^+$ : set of performance functions having constraint in the positive direction from nominal

$W^-$ : set of performance functions having constraint in the negative direction from nominal

$W_{ideal}$ : ideal value of  $W$  for a precision specification

$W_{max}$ : maximum value of  $W$  with process variations and without parasitics

- $W_{min}$ : minimum value of  $W$  with process variations without parasitics  
 $W_{nom}$ : nominal value of  $W$   
 $W_{prec}$ : the precision associated with  $W$   
 $W_{prec}^0$ : the precision associated with  $W$  without parasitics  
 $\Delta W_i$ : change in performance function  $W_i$  due to parasitics  
 $\Delta W_{prec}$ : degradation in precision of  $W$  caused by parasitics  
 $\Delta W_{max}^-$ : maximum allowed change in  $W$  in the negative direction due to parasitics  
 $\Delta W_{max}^+$ : maximum allowed change in  $W$  in the positive direction due to parasitics  
 $\Delta W_{proc}^-$ : maximum change in  $W$  in the negative direction due to process variations  
 $\Delta W_{proc}^+$ : maximum change in  $W$  in the positive direction due to process variations  
 $X$ : set of nodes in the circuit  
 $X_s$ : set of first nodes in the matched pairs of nodes  
 $\overline{X}_s$ : set of second nodes in the matched pairs of nodes  
 $X_q$ : set of unmatched nodes

## 4.2 Performance Constraints

Before formally defining the constraint-generation problem in Section 4.3, a description of performance constraints and their specification is provided to allow a better understanding of the succeeding material presented in the next sections.

A brief description of performance functions of analog circuits was provided in Section 2.1. In this thesis, performance constraints during layout design are defined to be the maximum variations (because of the interconnect parasitics) allowed in the performance functions from their nominal values to satisfy a set of performance specifications. The nominal value  $W_{nom}$  of a performance function  $W$  is defined as the performance obtained with no interconnect parasitic and process variation. Let  $\Delta W_{max}^+$  and  $\Delta W_{max}^-$  respectively denote the maximum positive and negative changes allowed in  $W$  from  $W_{nom}$  to satisfy the specifications on  $W$ .  $\Delta W_{max}^+$  and  $\Delta W_{max}^-$  represent the performance constraints associated with  $W$ .

Let  $W^+$  and  $W^-$  respectively denote the sets of performance functions, which have constraints in the positive and negative directions from their nominal values. Each performance function belonging to  $W^+$  will have a positive finite value for  $\Delta W_{i,max}^+$  and

each performance function belonging to  $W^-$  will have a positive finite value for  $\Delta W_{i\_max}$ . If a performance function has a two-sided constraint from its nominal, it will belong to both  $W^+$  and  $W^-$ . If a performance function has one sided constraint, it will belong to only one of the two sets.

Performance specifications of analog circuits can be categorized into two classes (a) Nonprecision Specifications and (b) Precision Specifications. The manner in which the performance constraints should be specified for a performance function depends on whether the associated specification is a nonprecision or a precision specification. Specification of performance constraints for each of these two cases is now described.

#### 4.2.1 Nonprecision Specifications

A non-precision specification allows the performance function to vary over a wide range, the most commonly encountered ones being those having one-sided performance constraints (the function is required to be only larger than some minimum  $W_{min}$ , or smaller than some maximum value  $W_{max}$ ). Examples of such specifications are those associated with bandwidth and phase margin of an opamp to be used in a custom design. Both bandwidth and phase margin will belong to  $W^-$  as there will be constraint on the maximum decrease in each of these performance functions. Often for such specifications, it is possible to overdesign the circuit, allowing substantial margins for performance degradation due to interconnect parasitics (which are not known during electrical design), and process variations. However, overdesign may require sacrifice of objective functions like chip area. Hence, it should only be used as means to compensate for factors such as process variations, over which the designer does not have any control, but whose effects can be statistically predicted in advance. Thus, for optimal design, it is a good idea to make the performance degradation caused by interconnect parasitics small compared to that caused by process variations.

#### 4.2.2 Precision Specifications

A precision specification is one which requires a performance function  $W$  to stay in a very narrow range around an ideal value  $W_{ideal}$  (e.g. the precision closed-loop gain of an opamp). Hence such performance functions belong to both  $W^+$  and  $W^-$ . Because of



the tight two-sided constraint on the performance, there is no room for overdesign.

The nominal value  $W_{nom}$  ( $W$  without interconnect parasitics and process variations) will be different from  $W_{ideal}$  because of the various nonidealities associated with the circuit. For example, in the circuit shown in Fig. 1.2, the nominal value of closed-loop gain of the circuit which has a precision specification associated with it, will deviate from its ideal value set by nominal ratio of capacitors  $C_1$  and  $C_2$ , due to the finite open-loop gain of the opamp.

In addition to the nonidealities, further change in performance functions will occur due to process variations and interconnect parasitics. The precision  $W_{prec}$  associated with the performance function is defined as its maximum deviation from the ideal value. i.e.

$$W_{prec} = \max(|W - W_{ideal}|) \quad (4.1)$$

$W_{prec}$  is always positive, and the lower its value, the higher is the accuracy realized. Let  $W_{prec}^0$  denote the precision of the performance function in the absence of interconnect parasitics. The critical interconnect parasitics which impair this nominal precision  $W_{prec}^0$  can be controlled very well, if proper care is taken during placement, routing and compaction. Hence, it is important to keep the degradation in precision caused by these layout phases (denoted by  $\Delta W_{prec}$ ) small compared to  $W_{prec}^0$ .

In order to compute the nominal precision, one should be able to estimate worst-case change in performance function due to process variations. The maximum changes in performance  $W$  (due to process variations) about its nominal value  $W_{nom}$  in the positive and negative directions are denoted by  $\Delta W_{proc}^+$  and  $\Delta W_{proc}^-$  respectively. Let  $W$  depend on  $N_q$  process parameters, say  $q_1, \dots, q_{N_q}$ , and each process parameter  $q_j$  has a maximum variation of  $\pm \Delta q_{j\_max}$  around its nominal value. If the performance function varies monotonically with respect to each  $q_j$  in the range described by process variations, a fast way of estimating  $\Delta W_{proc}^+$  and  $\Delta W_{proc}^-$  is to first compute sensitivities of performance with respect to the process parameters at their nominal values. Then, based on the signs of the individual sensitivities, the value of each process parameter can be set at one of its extreme values and the circuit simulated for a worst-case analysis. For precision specifications, since the performance is made to depend on parameters which are tightly controlled (such as ratios of capacitors or resistors), even linear analysis using sensitivities should yield a good estimate of the worst-case variation in performance. In that case, a simple expression for  $\Delta W_{proc}^+$

and  $\Delta W_{proc}^-$  is given by

$$\Delta W_{proc}^+ = \Delta W_{proc}^- = \sum_{j=1}^{N_q} |S_j| \Delta q_{j\_max} \quad (4.2)$$

where  $S_j$  is the sensitivity of  $W$  with respect to  $q_j$ . However, if linear approximations are not valid,  $\Delta W_{proc}^+$  and  $\Delta W_{proc}^-$  may not be equal.

The nominal precision  $W_{prec}^0$  of the performance can be computed in terms of the defined quantities as depicted by an example in Fig. 4.1. When all the routing parasitics are zero, the performance function  $W$  lies in the range  $(W_{min}, W_{max})$  as a result of process variations, where

$$W_{max} = W_{nom} + \Delta W_{proc}^+ \quad (4.3)$$

$$W_{min} = W_{nom} - \Delta W_{proc}^- \quad (4.4)$$

The maximum deviations of  $W$  from  $W_{ideal}$  in positive and negative directions are  $W_{max} - W_{ideal}$ , and  $W_{ideal} - W_{min}$  respectively. Hence, from definition (4.1),

$$W_{prec}^0 = \max(W_{max} - W_{ideal}, W_{ideal} - W_{min}) \quad (4.5)$$

$$= \max(W_{nom} - W_{ideal} + \Delta W_{proc}^+, \quad (4.6)$$

$$W_{ideal} - W_{nom} + \Delta W_{proc}^-)$$

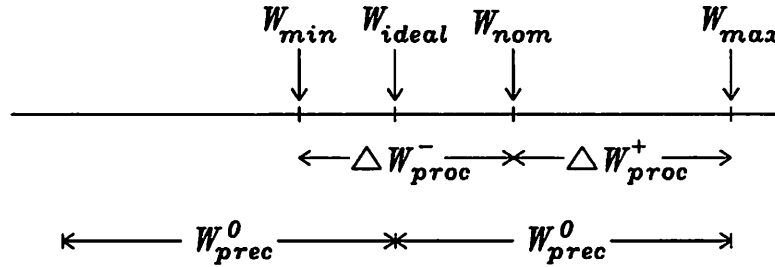


Figure 4.1: Parameters determining nominal precision.

If degradation in precision due to the parasitics should be limited to  $\Delta W_{prec} = r W_{prec}^0$ ,  $r \ll 1$ , then the performance function with the parasitics should stay in the range  $(W_{ideal} - (1+r)W_{prec}^0, W_{ideal} + (1+r)W_{prec}^0)$ . It is assumed that the relative process variations  $\Delta W_{proc}^+$  and  $\Delta W_{proc}^-$  do not change significantly because of shift in the operating point of the performance due to the parasitics. This should be a reasonable assumption, since in this analysis, one is interested in small shifts caused while the parasitics meet the performance

constraints. Also, the process variations in the parasitics themselves are taken care of by using worst-case sensitivities for modeling performance constraints (Section 4.4). Thus, the maximum positive and negative deviations ( $\Delta W_{max}^+$  and  $\Delta W_{max}^-$ ) in performance  $W$  from its nominal value because of parasitics should satisfy the following relations.

$$W_{nom} + \Delta W_{max}^+ + \Delta W_{proc}^+ = W_{ideal} + (1 + r)W_{prec}^0 \quad (4.7)$$

$$W_{nom} - \Delta W_{max}^- - \Delta W_{proc}^- = W_{ideal} - (1 + r)W_{prec}^0 \quad (4.8)$$

from which the general expressions for  $\Delta W_{max}^+$  and  $\Delta W_{max}^-$  are obtained as follows

$$\Delta W_{max}^+ = (1 + r)W_{prec}^0 - (W_{nom} - W_{ideal}) - \Delta W_{proc}^+ \quad (4.9)$$

$$\Delta W_{max}^- = (1 + r)W_{prec}^0 - (W_{ideal} - W_{nom}) - \Delta W_{proc}^- \quad (4.10)$$

As seen from (4.2),  $\Delta W_{proc}^+ = \Delta W_{proc}^-$ , when linear analysis is used for predicting these values. For this useful practical case, simple expressions will be derived for performance constraints  $\Delta W_{max}^+$  and  $\Delta W_{max}^-$ .

**CASE 1:**  $W_{nom} \geq W_{ideal}$

From (4.6)

$$W_{prec}^0 = W_{nom} - W_{ideal} + \Delta W_{proc}^+ \quad (4.11)$$

Substituting this expression in (4.9) and (4.10), and identifying  $rW_{prec}^0$  as  $\Delta W_{prec}$ , one gets

$$\Delta W_{max}^+ = \Delta W_{prec} \quad (4.12)$$

$$\Delta W_{max}^- = \Delta W_{prec} + 2(W_{nom} - W_{ideal}) \quad (4.13)$$

**CASE 2:**  $W_{nom} < W_{ideal}$

Again, from (4.6)

$$W_{prec}^0 = W_{ideal} - W_{nom} + \Delta W_{proc}^- \quad (4.14)$$

and substituting this in (4.9) and (4.10), one gets

$$\Delta W_{max}^+ = \Delta W_{prec} + 2(W_{ideal} - W_{nom}) \quad (4.15)$$

$$\Delta W_{max}^- = \Delta W_{prec} \quad (4.16)$$

To summarize, if linear analysis is valid for studying process variations of a precision function, then equations (4.12), (4.13), (4.15) and (4.16) can be used for specifying performance constraints. The maximum allowed change in a precision performance due to parasitics is (a)  $\Delta W_{prec}$  in the *offset direction* of  $W_{nom}$  with respect to  $W_{ideal}$ , and (b) an additional amount of twice this offset in the opposite direction, where  $\Delta W_{prec}$  (degradation in precision which can be tolerated) should be chosen small compared to the nominal precision of the circuit given by (4.11) and (4.14). When linear analysis is not valid, equations (4.6), (4.9) and (4.10) have to be used.

### 4.3 The parasitic constraint-generation problem

Parasitic constraint generation is the process of generating parasitic constraints for the layout tools from the high-level performance constraints of the circuit. If these parasitic constraints are satisfied by the constraint-driven layout tools, then the performance specifications of the circuit are automatically met after layout. This will reduce the need for expensive layout iterations as mentioned earlier in the introductory chapter. The parasitic constraint generation is formally defined as:

For a set of performance functions  $\{W_i\}$ ,  $i = 1, \dots, N_w$  and a set of parasitics  $\{p_j\}$ ,  $j = 1, \dots, N_p$ , *parasitic constraint generation*, is defined as generation of the following two types of constraints on a subset of  $\{p_j\}$ .

$$p_j = p_k \text{ (matching constraint)} \quad (4.17)$$

$$p_j \leq p_{j\_bound} \text{ (bounding constraint)} \quad (4.18)$$

to ensure that

$$\Delta W_i \leq \Delta W_{i\_max}^+, \quad \forall W_i \in W^+ \quad (4.19)$$

$$\Delta W_i \geq -\Delta W_{i\_max}^-, \quad \forall W_i \in W^- \quad (4.20)$$

Matching constraints are necessary to match parasitics associates with certain pairs of nodes and branches (necessary in differential circuits and circuits employing matched devices). Bounding constraints limit the maximum values of the parasitics to meet the performance constraints. Constraints are imposed only on a subset of the parasitics as others may be noncritical.

## 4.4 Modeling Performance Constraints

In general, the parasitic constraint-generation problem seems to be extremely difficult, as the performance functions are nonlinear functions of the parasitics and no analytical expressions for those functions may be available. However, since one wants to limit variations in performance functions to be small around their nominal values (because of the additional interconnect parasitics), linear approximations using sensitivities can be used to model dependence of performance on the parasitics, when the parasitics are within the bounds specified by the bounding constraints. It is to be emphasized that the performance degradation will not be automatically small, if parasitic constraints are not imposed.

As defined in Section 2.2 the sensitivity  $S_{ij}$  of a performance function  $W_i$  with respect to a parasitic  $p_j$  at the nominal value of  $W_i$  is:

$$S_{ij} = [\partial W_i / \partial p_j]_{p_j=0} \quad (4.21)$$

In case of matched parasitics, value of only one parasitic can be independently controlled. Hence, the matched pair is treated like a fictitious single parasitic for the generation of bounding constraints. The bounding constraint generated on that fictitious parasitic imposes the same constraint on both the parasitics in the matched pair. The expression for the sensitivity associated with that single fictitious parasitic is derived later in this section.

In general, for each performance function, some of the sensitivities will be positive and some negative. However, when the layout is not available, one cannot take advantage of the possible cancellation effects while generating bounds on the various parasitics, except for the special case when a matching constraint is imposed on a pair of parasitics (the effect matching is handled by defining the fictitious parasitic). Hence, for modeling the constraint on a performance function in positive (negative) direction, only the parasitics which have positive (negative) sensitivities with respect to that performance function are considered.

Approximations to performance constraints can thus be modeled by the following inequalities.

$$\sum_{j=1}^{N_p} S_{ij}^+ p_j \leq \Delta W_{i\_max}^+ \quad \forall W_i \in W^+ \quad (4.22)$$

$$\sum_{j=1}^{N_p} S_{ij}^- p_j \leq \Delta W_{i\_max}^- \quad \forall W_i \in W^- \quad (4.23)$$

where

$$S_{ij}^{\pm} = S_{ij}, \text{ if } S_{ij} \geq 0; S_{ij}^{\pm} = 0, \text{ if } S_{ij} < 0 \quad (4.24)$$

$$S_{ij}^{\bar{\pm}} = -S_{ij}, \text{ if } S_{ij} \leq 0; S_{ij}^{\bar{\pm}} = 0, \text{ if } S_{ij} > 0 \quad (4.25)$$

Note that  $S_{ij}^{\pm}$  and  $S_{ij}^{\bar{\pm}}$  are always nonnegative.

Sensitivity computation for matched parasitics is now addressed. Let two parasitics  $p_j$  and  $p_k$  are nominally matched, such that  $p_j = p_k$ , and process variations are ignored. For the purpose of generation of bounding constraints, the two parasitics are replaced by a single fictitious parasitic  $p_0$  whose value is same as the nominal values of  $p_j$  and  $p_k$  (which are same without any process variations) .

Then the sensitivity of a performance function  $W_i$  with respect to  $p_0$  is

$$S_0 = [\partial W_i / \partial p_j][\partial p_j / \partial p_0] + [\partial W_i / \partial p_k][\partial p_k / \partial p_0] \quad (4.26)$$

$\partial p_j / \partial p_0$  and  $\partial p_k / \partial p_0$  are both unity since  $p_j = p_0$  and  $p_k = p_0$ . Hence,

$$S_0 = S_{ij} + S_{ik} \quad (4.27)$$

However, due to finite *mismatch* because of process variations, use of this expression for sensitivity may give meaningless results, particularly when  $S_{ij}$  and  $S_{ik}$  tend to cancel each other.

Expressions for the upper and lower bounds of sensitivity have been derived in Appendix 4.10 taking worst-case process variations into account, and are listed below.

CASE I:  $S_{ij}$  and  $S_{ik}$  are of the same sign

$$(1 \pm r_{max})S_0 \quad (4.28)$$

CASE II:  $S_{ij}$  and  $S_{ik}$  are of opposite signs

$$S_0 \pm [r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}|], \text{ if } |S_{ij}| \leq |S_{ik}| \quad (4.29)$$

$$S_0 \pm [r_{max}(|S_{ij}| - |S_{ik}|) + r_{mis}|S_{ik}|], \text{ if } |S_{ij}| > |S_{ik}| \quad (4.30)$$

where  $r_{max}$  is the maximum fractional process variation of a parasitic, and  $r_{mis}$  is the maximum fractional mismatch between the matched parasitics.

For the important special case when  $S_{ij} = -S_{ik}$  (refer to Appendix 4.10 for an example where this situation may arise), the bounds on the sensitivity become  $\pm r_{mis}|S_{ij}|$

and  $-r_{mis}|S_{ij}|$ . This is compared to "0" sensitivity predicted if mismatch is not taken into account. The magnitude of these bounds can be significant when  $|S_{ij}|$  is large, since  $r_{mis}$  can be as large as 0.2 (as the values of interconnect parasitics are poorly controlled).

Even for parasitics which are not matched, process variations should be considered to define the worst-case sensitivities. The worst-case sensitivity with respect to the nominal value of a single unmatched parasitic  $p_j$  is  $S_{ij} = (1 + r_{max})S_{ij_0}$ , where  $S_{ij_0}$  is the sensitivity without any process variations (easy to show).

Let the upper bound on the sensitivity  $S$  is denoted by  $S_{up}$ , and the lower bound by  $S_{lo}$ . Then, for a worst-case modeling of performance constraints, following substitutions have to be made for  $S_{ij}$  in (4.22) and (4.23).

$$S_{ij} = S_{up} \text{ in (4.22), } S_{ij} = S_{lo} \text{ in (4.23)} \quad (4.31)$$

## 4.5 Matching Constraints on Parasitics

In this section, derivation of capacitive matching constraints from matched-node-pair information in differential circuits will be shown (inductive and resistive matching constraints can be derived in a similar manner from matched-branch-pair information).

Let  $X$  denote the set of nodes present in a circuit. In each differential circuit, there exists a set of matched pairs of nodes

$$M = \{(n_i, \bar{n}_i), i = 1, \dots, N_m\} \quad n_i, \bar{n}_i \in X \quad (4.32)$$

The sets of nodes  $\{n_i\}, \{\bar{n}_i\}$  are denoted by  $X_s$  and  $\bar{X}_s$  respectively. The set of remaining nodes which are not matched is denoted by  $X_q$ . The interconnect capacitance present from node  $n_i$  to ground is denoted by  $C(n_i)$  and that between nodes  $n_i$  and  $n_j$  by  $C(n_i, n_j)$ .

Usually, the pair of nodes in each matched pair of a differential circuit have equal impedance levels, and equal and opposite transfer functions to the output. For impedance matching, the capacitances to ground associated with each matched pair of nets should be equal. Also, when a net which is not matched, comes close to a matched pair of nets in the layout, it is required to match the coupling capacitances between that net and the pair of matched nets. This causes equal loading on the pair of nets, and in addition, any noise

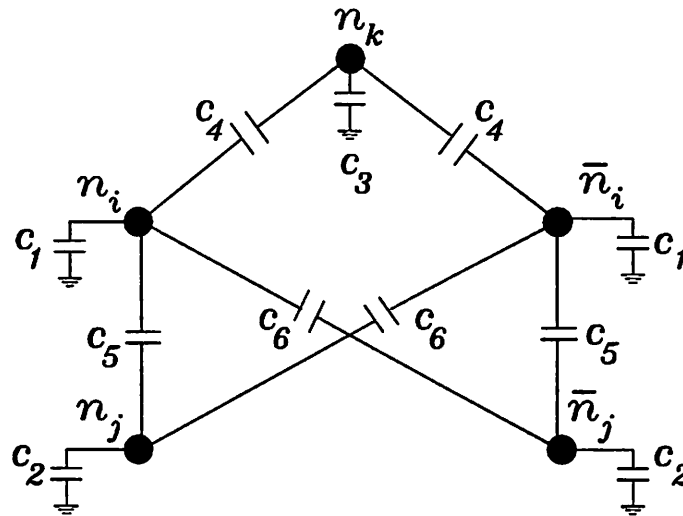


Figure 4.2: Matching constraints.

in the third net is equally coupled to the pair resulting in noise cancellation at the output. When *two pairs* of matched nets come close to each other, it is necessary to match the direct-coupling capacitances ( $C_5$  in Fig. 4.2) and cross-coupling capacitances ( $C_6$  in Fig. 4.2). Besides causing symmetrical loading, this ensures that equal level of noise on the two nodes of one matched pair causes the same on the other pair if any coupling is present. The capacitive constraints can thus be derived from matched-node-pair information of the circuit using the following relations.

**Algorithm 1 (Derivation of Matching Constraints)**

$$C(n_i) = C(\bar{n}_i), \quad \forall n_i \in X_s \quad (4.33)$$

$$C(n_i, n_k) = C(\bar{n}_i, n_k), \quad \forall n_i \in X_s, \forall n_k \in X_q \quad (4.34)$$

$$C(n_i, \bar{n}_j) = C(\bar{n}_i, n_j), \quad \forall n_i, n_j \in X_s \quad (4.35)$$

$$C(n_i, n_j) = C(\bar{n}_i, \bar{n}_j), \quad \forall n_i, n_j \in X_s \quad (4.36)$$

Matching constraints have been pictorially illustrated in Fig. 4.2, where capacitances to be matched have the same label. The matching constraints which are derived from Algorithm 1, not only impose the corresponding symmetry requirements on the layout tool, but also affect the worst-case modeling of performance constraints as described in Section 4.4.



## 4.6 Bounding Constraints on Parasitics

Although the actual bounding constraints are on performance functions represented by (4.22) and (4.23), it may be difficult for some layout tools to handle them directly. For example, at any stage of routing, one does not have a global picture of all the parasitics affecting the various performance functions. On the other hand, it is much more convenient for the router to meet bounding constraints on the individual parasitics which are derived from (4.22) and (4.23). The process for obtaining the bounding constraints on individual parasitics from the performance constraints will now be described.

### 4.6.1 Limits for Parasitics

During layout design, often it is possible to estimate reasonable limits (not to be confused with bounding constraints) for the various parasitics. For example, for a given placement, the minimum possible values of capacitances for various nets are determined by the half perimeters of bounding boxes associated with the pins. When the routing algorithm to be used is known, then it may be possible to estimate lower and upper limits for coupling capacitances also. Algorithms for estimating such limits for the constraint-driven channel router ART described in Chapter 5, have been presented in Appendix 4.11. When such limits are estimated for a given placement, they can be used for generating bounding constraints on the parasitics for routing as described later. When constraints have to be generated for placement, only very rough estimates for limits on the parasitics can be estimated, based on the estimated dimensions of the chip.

### 4.6.2 Selecting the Critical Parasitics

If  $N_{mc}$  matching constraints are imposed, then there are  $(N_p - N_{mc})$  *independent* parasitics available, on which bounding constraints can be imposed. However, in practice a significant fraction of these parasitics will have very low sensitivities. If constraints are generated on all the independent parasitics, a large number of noncritical constraints will unnecessarily overload the layout tools. For any given circuit, the sensitivities of each performance function with respect to the parasitics follow a particular distribution from

the high to the low end. The nature of this distribution depends on the circuit and the performance function considered. Determination of a cutoff point for selecting the critical parasitics will now be discussed.

Initially, sensitivity computation has to be performed with respect to all the parasitics which may possibly exist in the layout. For example, if capacitances are considered then capacitances to ground from all nodes, and coupling capacitances between all possible pairs of nodes will be considered. If the *adjoint* technique of sensitivity analysis[42] is used, the analysis time will be almost independent of the number of parasitics considered.

For each performance function  $W_i \in W^+$ , let the parasitics with nonnegative sensitivities be sorted in increasing value, and for each  $W_i \in W^-$ , let the negative sensitivities be sorted in increasing magnitude. Suppose  $p_{j\_max}$  is the conservative estimate of the maximum value that  $p_j$  can attain in the layout (as discussed in Section 4.6.1). Note that the order of sorting may be different for different performance functions. If only performance function  $W_i$  is considered, then the first  $N_i^+$  and  $N_i^-$  parasitics in the sorted lists which respectively make positive and negative contribution to  $W_i$  can be considered noncritical, if  $N_i^+$  and  $N_i^-$  are the largest integers such that

$$\sum_{j=1}^{N_i^+} S_{ij}^+ p_{j\_max} \leq \alpha \Delta W_{i\_max}^+ \quad \text{if } W_i \in W^+ \quad (4.37)$$

$$\sum_{j=1}^{N_i^-} S_{ij}^- p_{j\_max} \leq \alpha \Delta W_{i\_max}^- \quad \text{if } W_i \in W^- \quad (4.38)$$

where the threshold  $\alpha$  should be set to a value small compared to 1, say 0.01. Effective values of  $\Delta W_{i\_max}^+$  and  $\Delta W_{i\_max}^-$  can then be reduced by the same fraction. Hence, this procedure separates the parasitics which can collectively only cause a performance degradation small compared to the maximum allowed. Let  $R_i$  denote the intersection of sets of these  $N_i^+$  and  $N_i^-$  parasitics. The set of parasitics which can be considered noncritical when all the performance functions are considered is given by

$$R = \bigcap_{i=1}^{N_w} R_i \quad (4.39)$$

The noncritical parasitics so determined are eliminated from further analysis. Let the remaining *critical* parasitics be indexed from 1 to  $N_{cp}$  (this indexing will be used in Sections 4.6.3 and 4.6.4). If parasitics which have different dimensions (capacitances, resistances and inductances) are considered, then different sorted lists have to be maintained

for each kind (since the sensitivities are also of different dimensions), and elimination carried out separately. But after this elimination, the critical parasitics can all be considered together for the generation of bounding constraints.

Even for the critical parasitics chosen for further consideration, if the value of a parasitic is so small that it falls below a predefined threshold, it can be considered to be zero. Hence, during final extraction (used for layout verification), one can avoid overloading the netlist by ignoring such small parasitics. The threshold value  $p_{thresh}$  of the parasitic of a given kind, say capacitance, can be estimated by solving for the value of the parasitic, for which the sum of the contributing terms in LHS of equations (4.37) and (4.38) satisfy the RHS for a given value of  $\alpha$ . The value of  $\alpha$  is set, and corrected for, as described before. Note that  $p_{thresh}$  is different from (usually much less than) the bounding constraint  $p_{j\_bound}$  associated with a parasitic  $p_j$ .

### 4.6.3 Modeling Layout Flexibility

In general, there are several parasitics affecting several performance functions. Hence, there are several possible combinations of bounding constraints on parasitics which will satisfy the performance constraints. However, they may not all be equally easy for layout tools to meet. This is particularly the situation when the parasitics have substantially different sensitivities. In that case, given one set of constraints it may be possible to *tighten* the constraints on *more sensitive* parasitics by *small* amounts and *relax* the constraints on *less sensitive* ones by *large* amounts to obtain another set of constraints easier to meet. As an example, let there be two parasitics  $p_1$  and  $p_2$  and one performance function  $W$  of interest. Let  $W$  be ten times more sensitive to  $p_2$  than to  $p_1$ . Suppose  $p_1 \leq 1fF$  and  $p_2 \leq 1fF$  form one possible set of bounding constraints which satisfy the specified constraint on the performance. Then if the bound on  $p_1$  is reduced by a small amount (say  $0.1fF$ ), the bound on  $p_2$  can be relaxed by a much larger amount (about  $1fF$ ), since  $p_1$  is ten times more sensitive than  $p_2$ . The second set of constraints may be easier to meet as it increases the total allowable capacitance during layout design.

The general problem is formalized by introducing the notion of *layout flexibility* associated with the bounding constraints, and then maximizing this flexibility subject to the performance constraints while generating the bounding constraints on the parasitics.

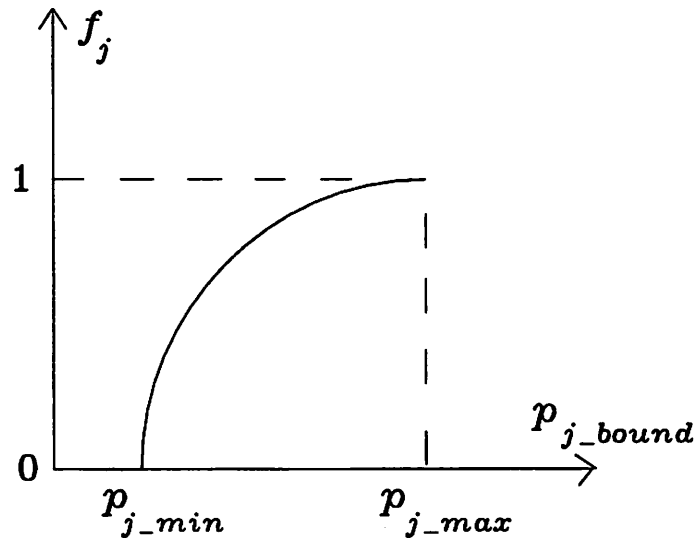


Figure 4.3: Flexibility model.

As described in Section 4.6.1, often it is possible to make estimates of maximum and minimum values ( $p_{j\_min}$  and  $p_{j\_max}$ ) a parasitic  $p_j$  can attain in the layout. Such information can be used to model the layout flexibility associated with the parasitic constraints. A bounding constraint  $p_{j\_bound}$  should not be less than  $p_{j\_min}$ , otherwise that constraint can not be met. Also, the constraint should not be pushed above  $p_{j\_max}$  (the maximum value which the parasitic can attain in the layout). The reason for this being that bounding constraints are generated by maximizing overall layout flexibility. If the constraint is allowed to increase above  $p_{j\_max}$ , no additional flexibility is provided to the tool, but it might require unnecessary tightening of constraint on some other parasitic (possible when the two parasitics have different sensitivities, as illustrated earlier). Hence, each bounding constraint should satisfy

$$p_{j\_min} \leq p_{j\_bound} \leq p_{j\_max} \quad (4.40)$$

The total layout flexibility is modeled as sum of flexibilities associated with each bounding constraint.  $f_j$  is defined as the layout flexibility associated with bounding constraint on  $p_j$ , and is assumed to be a number ranging from 0 (no flexibility), to 1 (full flexibility). Then,  $f_j$  should be zero when  $p_{j\_bound} = p_{j\_min}$ ; and it should be unity when  $p_{j\_bound} = p_{j\_max}$ . Also, when  $p_{j\_bound}$  increases beyond  $p_{j\_max}$ , since no additional flexibility is given to the layout tool, the slope of the  $f_j$  vs.  $p_{j\_bound}$  curve should be zero at  $p_j = p_{j\_max}$ , if one wants to fit a smooth curve. Since one cannot use a linear relationship

to model this, a second order polynomial as depicted in Fig. 4.3 is used.

It can be shown that the second order polynomial in  $p_{j\_bound}$  satisfying the above three conditions is

$$f_j = a_j + b_j p_{j\_bound} + c_j p_{j\_bound}^2 \quad (4.41)$$

where

$$c_j = -1/(p_{j\_max} - p_{j\_min})^2 \quad (4.42)$$

$$b_j = -2c_j p_{j\_max} \quad (4.43)$$

$$a_j = -b_j p_{j\_min} - c_j p_{j\_min}^2 \quad (4.44)$$

The total layout flexibility is

$$f = \sum_1^{N_{cp}} f_j \quad (4.45)$$

$$= \sum_1^{N_{cp}} a_j + \sum_1^{N_{cp}} b_j p_{j\_bound} + \sum_1^{N_{cp}} c_j p_{j\_bound}^2 \quad (4.46)$$

#### 4.6.4 The Main Algorithm

The problem of maximizing the total flexibility can be converted into the standard form of minimization by changing the sign of the objective function (denoted by *cost*). Hence,

$$cost = -f \quad (4.47)$$

$$= -\sum_1^{N_{cp}} a_j - \sum_1^{N_{cp}} b_j p_{j\_bound} - \sum_1^{N_{cp}} c_j p_{j\_bound}^2 \quad (4.48)$$

For minimization, the first term in (4.48) which is a constant can be ignored. Hence, the cost function can be more compactly written as  $p^T Q p + b^T p$ , where

$p$  is an  $N_{cp} \times 1$  column vector having  $j$ th element equal to  $p_{j\_bound}$ .

$Q$  is an  $N_{cp} \times N_{cp}$  positive-definite diagonal matrix whose  $j$ th diagonal element is  $-c_j$  ( $c_j < 0$ ).

$b$  is an  $N_{cp} \times 1$  column vector whose  $j$ th element is  $-b_j$ .

The minimization has to be subject to the linear performance constraints represented by (4.22) and (4.23) involving the *critical* parasitics (refer to Section 4.6.2), and the imposed limit  $(p_{min}, p_{max})$ , where  $p_{min}$  and  $p_{max}$  are the vectors of the minimum and maximum estimated values of the individual parasitics. The main algorithm for generation of bounding constraints now follows:

**Algorithm 2 (Generation of Bounding Constraints)**

$$\text{minimize } p^T Q p + b^T p \quad (4.49)$$

such that

$$S_i^+ p \leq \Delta W_{i\_max}^+ \quad \forall W_i \in W^+ \quad (4.50)$$

$$S_i^- p \leq \Delta W_{i\_max}^- \quad \forall W_i \in W^- \quad (4.51)$$

$$p_{min} \leq p \leq p_{max} \quad (4.52)$$

where  $S_i^+$  and  $S_i^-$  are  $1 \times N_{cp}$  row vectors whose  $j$ th elements are  $S_{ij}^+$  and  $S_{ij}^-$  respectively.

This is a quadratic programming problem (referred to as the *constraint-generation problem* from now on), in  $N_{cp}$  variables, with a quadratic cost function and linear constraints. Due to positive definiteness of  $Q$ , it can be solved by standard QP packages. By definition, a *feasible solution* of this problem is a solution which satisfies (4.50), (4.51) and (4.52). An *optimum solution*  $p_{opt}$  is a feasible solution which minimizes the cost function given by (4.49). A vector  $p_c$  is called a *bounding constraint*, if for all  $p \leq p_c$  (where this inequality implies term by term inequality for the two vectors), the performance constraints ((4.50) and (4.51)) are satisfied.

**Lemma 4.1** *A vector  $p_c$  is a bounding constraint if and only if it satisfies (4.50) and (4.51).*

**Proof:** If  $p_c$  is a bounding constraint, by definition, any  $p \leq p_c$  satisfies the performance constraints described by (4.50) and (4.51), and so does  $p_c$ . To show the reverse, it is observed that the coefficients and RHS of (4.50) and (4.51) are all nonnegative (refer to Sections 4.3 and 4.4). So if a solution is found which meets performance constraints (4.50) and (4.51), then the parasitics whose values are less than those of the solution vector also meet the performance constraints.

**Corollary 4.1** *A feasible solution of the constraint-generation problem is a bounding constraint, but the reverse is not necessarily true (directly follows from Lemma 4.1).*

**Theorem 4.1** *The necessary and sufficient condition for finding a feasible solution of the constraint-generation problem is that  $p_{min}$  be a feasible solution.*

It is obviously a sufficient condition, since it is possible to find at least one feasible solution, i.e.  $p_{min}$ . Using the argument of proof in Lemma 4.1, it can be easily shown that if  $p_{sol}$  is any feasible solution,  $p_{min}$  is also a feasible solution and hence a necessary condition for finding a feasible solution.

**Lemma 4.2** *There is a unique optimum solution  $p_{opt}$  for the constraint-generation problem.*

This follows from the properties of quadratic programming, and from the fact that  $Q$  matrix is strictly positive definite.

**Theorem 4.2** *The optimum solution  $p_{opt}$  of the constraint-generation problem yields a maximal bounding constraint in the closed interval  $[p_{min}, p_{max}]$ , i.e. for no other bounding constraint  $p_c$  in this interval,  $p_c \geq p_{opt}$ .*

**Proof:** The result will be shown for a more general case, when  $f_j$  is a strictly monotonically increasing function of  $p_{j\_bound}$  in the closed interval  $[p_{j\_min}, p_{j\_max}]$  (the flexibility model which is used for  $f_j$  satisfies this condition). Suppose there exists a bounding constraint  $p_c$  ( $p_{min} \leq p_c \leq p_{max}$ ), which is different from  $p_{opt}$ , and  $p_c \geq p_{opt}$ .  $p_c$  is a feasible solution of the constraint generation problem, since it satisfies (4.50) and (4.51) (from Lemma 4.1), and also satisfies (4.52) according to the assumption made.

Using (4.47) and (4.45),

$$cost(p_c) - cost(p_{opt}) = \sum_1^{N_{cp}} [f_j(p_{j\_opt}) - f_j(p_{j\_c})] \quad (4.53)$$

where  $p_{j\_opt}$  and  $p_{j\_c}$  are the  $j$ th elements of  $p_{opt}$  and  $p_c$  respectively.

Since  $p_{opt} \leq p_c$ , and  $p_c$  is different from  $p_{opt}$ ,  $p_{j\_opt} < p_{j\_c}$  for at least one  $j$ . Due to the strictly monotonically increasing nature of  $f_j$ , it follows that

$$cost(p_c) < cost(p_{opt}) \quad (4.54)$$

This is a contradiction, since  $p_c$  is a feasible solution and cannot have a cost function less than that associated with the optimum solution  $p_{opt}$ .

The results of Lemma 4.1, Corollary 4.1 and Theorem 4.1 are valid irrespective of how the layout flexibility is modeled. As shown, Theorem 4.2 applies to any flexibility model for  $f_j$  which is strictly monotonically increasing with the bounding constraint in the desired range. The way this flexibility is modeled is going to determine how easily the generated bounding constraints can be met by the layout tool.

The constraint generation will be successful if  $p_{min}$  is a feasible solution. If the minimum possible values of some of the parasitics (like capacitance to ground) due to placement are such that a feasible solution does not exist for routing, then the placement has to be changed. Shielding can also be used to eliminate unavoidable coupling capacitances, and reduce the effective values of minimum capacitances to obtain a feasible solution, as discussed in Appendix 4.12.

The minimum estimate  $p_{j\_min}$  of a parasitic can always be set to threshold  $p_{thresh}$  (defined in Section 4.6.2) if it falls below  $p_{thresh}$ . Hence, the bounding constraint on any parasitic is always forced to lie above certain nonzero threshold. This makes sure that the layout tool has nonzero flexibility associated each critical parasitic, without affecting the chances of performance constraints being met (because of the way  $p_{thresh}$  is defined).

## 4.7 PARCAR

The algorithms which were presented are implemented in PARCAR, a parasitic constraint generator. It has approximately 7000 lines of C code. An overall flow diagram of PARCAR is shown in Fig. 4.4. It generates constraints on interconnect capacitances (both line-to-ground and line-to-line). It can either be used for generating constraints on routing (for a given placement) or for generating constraints on placement itself.

### 4.7.1 Program Input

The input data PARCAR needs are (a) circuit netlist, (b) names of matched-node pairs, (c) performance constraints. Besides these inputs, some additional inputs may be necessary depending on the mode the program is used in. They are: (a) placement information, (b) technology information, (c) maximum and minimum limits for parasitics.



The circuit netlist can be a SPICE netlist or a netlist for the switched-capacitor simulator swap[41]. For the circuit, the nodes which have to be matched have to be specified. The performance constraints are specified by first defining the performance functions of interest for the circuit. This can be done by choosing one or more among a set of standard specifications such as gain, bandwidth, phase margin etc. If a performance function which is not supported by the program is to be defined, then a special interface can be used[96] in which the performance functions are described procedurally. The constraints on the performance functions of interest are then specified as maximum allowed changes from their nominal values in either positive or negative or both directions (due to interconnect parasitics).

When the program is used to generate constraints on routing for a given placement, the placement and technology information are fed to the program. They are used to estimate maximum and minimum values for the individual parasitics which are used in the constraint generation. The technology information consists of the coefficients used for computation of line-to-ground and coupling capacitances between interconnects. These coefficients can be generated automatically by using the model generator CAPMOD described in Chapter 6. The technology information also consists of worst-case process variation of parasitics and worst-case mismatch between two matched parasitics (expressed in percent). When constraints have to be generated for placement, then conservative minimum and maximum estimates for all the parasitics can be specified simply based on the estimates of chip dimensions as mentioned in Section 4.6.1.

#### **4.7.2 Program Operation**

The program has two modes of operation (a) a normal mode and (b) a special mode. In the normal mode, PARCAR can be interfaced to any layout tool which can meet bounding and matching constraints on parasitics. A constraint-driven area router RoAD has been interfaced to PARCAR in this mode [93]. The placement information in this mode has to be provided in OCT format. A special mode has been developed in which the program has been interfaced to a constraint-driven channel router ART described in Chapter 5. In this mode, the user can directly input the dimensions of the unrouted channels, and the constraint generator generates constraints for each channel separately, as the channel router routes one channel at a time. A parasitic may have a contribution coming from more

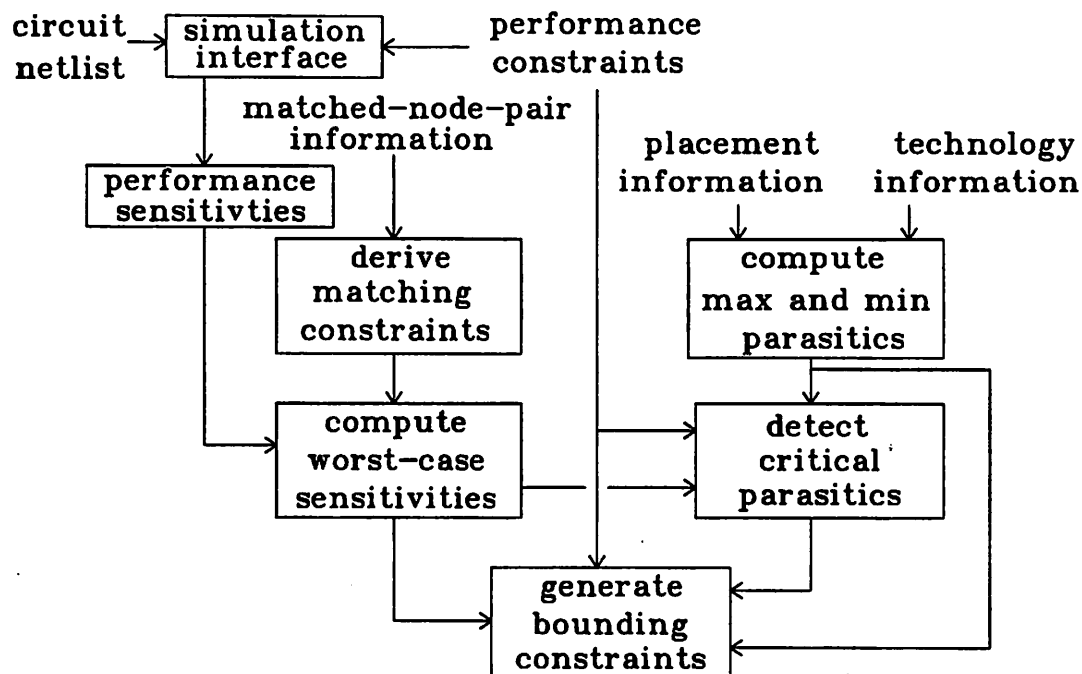


Figure 4.4: Flow diagram of PARCAR.

than one channel. For example if two nets run close to each other in more than one channel the coupling capacitance between the two nets is distributed over more than one channel. For the purpose of constraint generation, these contributions for a parasitic coming from different channels are treated as different parasitics (having the same performance sensitivities).

The program first inserts zero-valued capacitances from all the nodes to ground and between all possible pairs of nodes in the circuit. Interface has been developed for the simulators SPICE3[38][39] and SWAP[41], for accessing the sensitivities of the performance functions with respect to all the parasitics at zero nominal values. The sensitivities available from the circuit simulators are those associated with circuit variables (i.e. node voltages and branch currents). Intermediate calculations have to be performed as described in Section 2.2 to compute the sensitivities of the performance functions. The matched node pair information is then used to impose matching constraints on the parasitics and reduce the number of independent parasitics to be considered for generating bounding constraints. Then worst-case sensitivities with respect to the independent parasitics are computed as described in Section 4.4 using the technology information provided. When running in the

special mode with the constraint-driven channel router ART described in Chapter 5, the program automatically computes the maximum and minimum values of the parasitics in each channel. Details of this process are included in Appendix 4.11. When running in the normal mode with the constraint-driven area router RoAD[93], a special interface program is used to perform this task. The maximum values for parasitics are used along with the worst-case sensitivities to select the critical parasitics (Section 4.6.2). The bounding constraints on these critical parasitics are then generated using the algorithm presented in Section 4.6.4. A quadratic programming package “qpr2” is used for this purpose.

The output of the program consists of the critical pairs of nodes between which bounding constraints are generated, the associated bounds on the capacitances, and the associated performance sensitivities. Intermediate files are created to store the sensitivities of each performance function with respect to all the parasitics (critical and noncritical), in decreasing order. Hence the user can obtain information about the noncritical parasitics as well. For detailed information regarding the use of this program refer to Appendix A.

## 4.8 Results

### 4.8.1 SC Filter Example

Results of automatic constraint generation are now presented for a fifth-order switched-capacitor filter circuit as shown in Fig. 4.5. The filter circuit was simulated using the switch-cap simulator SWAP[41]. The simulated response has a nominal ripple of about  $+/- 0.1dB$  in the pass band of 0-1.6KHz. The nominal magnitude response assumes maximum positive and negative peak in the passband at about 0.1KHz, and 1.6KHz respectively. Hence, responses at these two frequencies have been considered to model the performance constraints, although more samples can be taken for better accuracy. The ideal value of magnitude response in the pass band has been taken as the value about which peak deviation in nominal response curve is symmetric. It is the relative flatness of response curve in the pass band which is the performance of interest in this example. After going through the procedure suggested in Section 4.2.2, in a manner similar to that in last example,  $\Delta W_{max}^+$  and  $\Delta W_{max}^-$  in the magnitude response were set to 0.01dB and -0.17dB at 0.1KHz, and vice versa at 1.6KHz. The worst-case process variation was calculated by

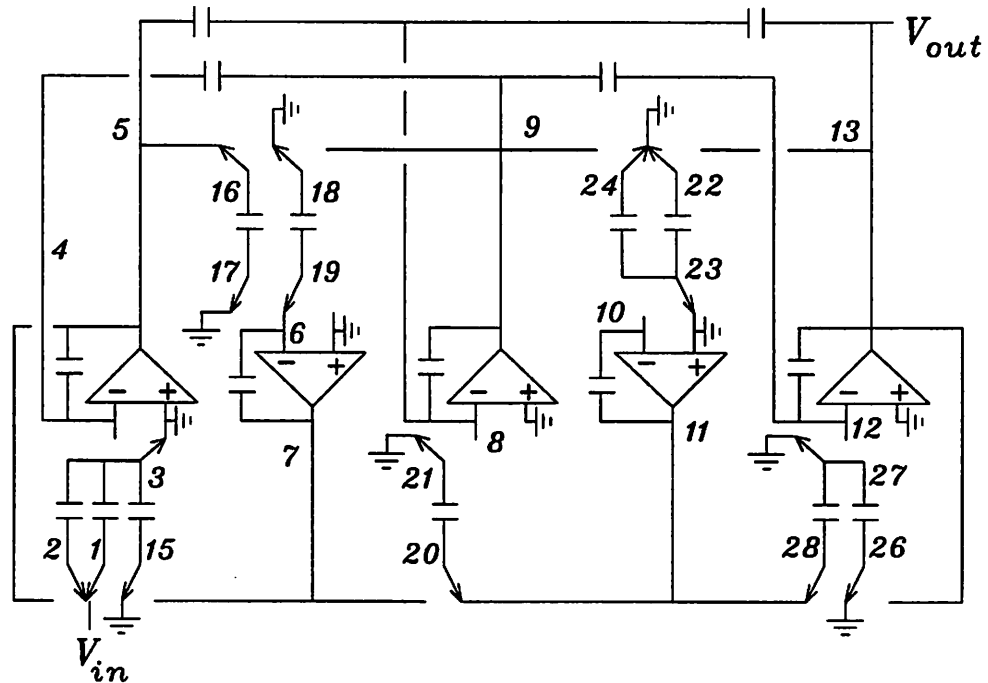


Figure 4.5: Fifth-order low-pass SC filter.

considering a maximum variation of 0.1% in the ratios between the integrating capacitor and the other capacitors connected to the input of each opamp (directly or through switches). The nominal frequency response of an SC filter is designed to depend on these ratios only, to the first order.

The placement and global routing for the layout of the filter was generated using the program ADORE[16]. The program follows a standard-cell type of placement. It places all the modules in three rows, switches in the top row, opamps in the bottom row, and capacitors in the middle row. There is a channel available for routing above each row. The switches are oriented in such a fashion that the top channel contains all the clock lines and no analog net. Hence, constraints have been generated for the other two channels only.

The circuit has 25 nodes. Sensitivity analysis was performed using SWAP. The sensitivities of magnitude of frequency response of the filter at two different 0.1 kHz and 1.6 kHz have been shown in the Table 1 and Table 2 respectively. Bounding constraints were generated on 30 coupling capacitances. Five of the constraints in one of the channels as well as the actual estimated capacitances (after the channel was routed by the constraint-driven channel router ART) have been shown in Table 3. Although two parasitic constraints could

not be met by the router, since the other constraints were met with some margins, the circuit with the extracted parasitics met the performance constraints. More detailed results for constraint-driven channel routing are provided in Chapter 5.

The CPU time for generating the constraints was about 5sec on VAX 8650. Error due to linear approximations (used in modeling performance constraint) was found to be less than 0.5%.

<i>net1</i>	<i>net2</i>	<i>sensitivity at 0.1kHz</i>
18	21	-0.0062 dB/fF
22	23	-0.0044 dB/fF
18	23	0.0044 dB/fF
16	19	-0.0043 dB/fF
17	18	0.0043 dB/fF

Table 1: Sensitivities of magnitude response at 0.1KHz w.r.t five of the most sensitive capacitances.

<i>net1</i>	<i>net2</i>	<i>sensitivity at 1.6kHz</i>
23	28	0.0184 dB/fF
18	21	-0.0156 dB/fF
18	23	0.0104 dB/fF
27	28	0.0062 dB/fF
17	18	0.0058 dB/fF

Table 2: Sensitivities of magnitude response at 1.6KHz w.r.t five of the most sensitive capacitances.

<i>net1</i>	<i>net2</i>	$C_{j\_bound}$	$C_{j\_actual}$
18	21	0.0015fF	0fF
12	28	3.3fF	1.3fF
4	15	0.01fF	0fF
27	28	0.5fF	0.01fF
23	28	0.0015fF	0.007fF

Table3: Five of the bounding constraints on coupling capacitances and the actual estimated capacitances after routing.

### 4.8.2 CMOS Opamp Example

This section presents the results of automatic constraint generation for a CMOS fully-differential folded-cascode opamp. The schematic of the gain stage of the opamp is

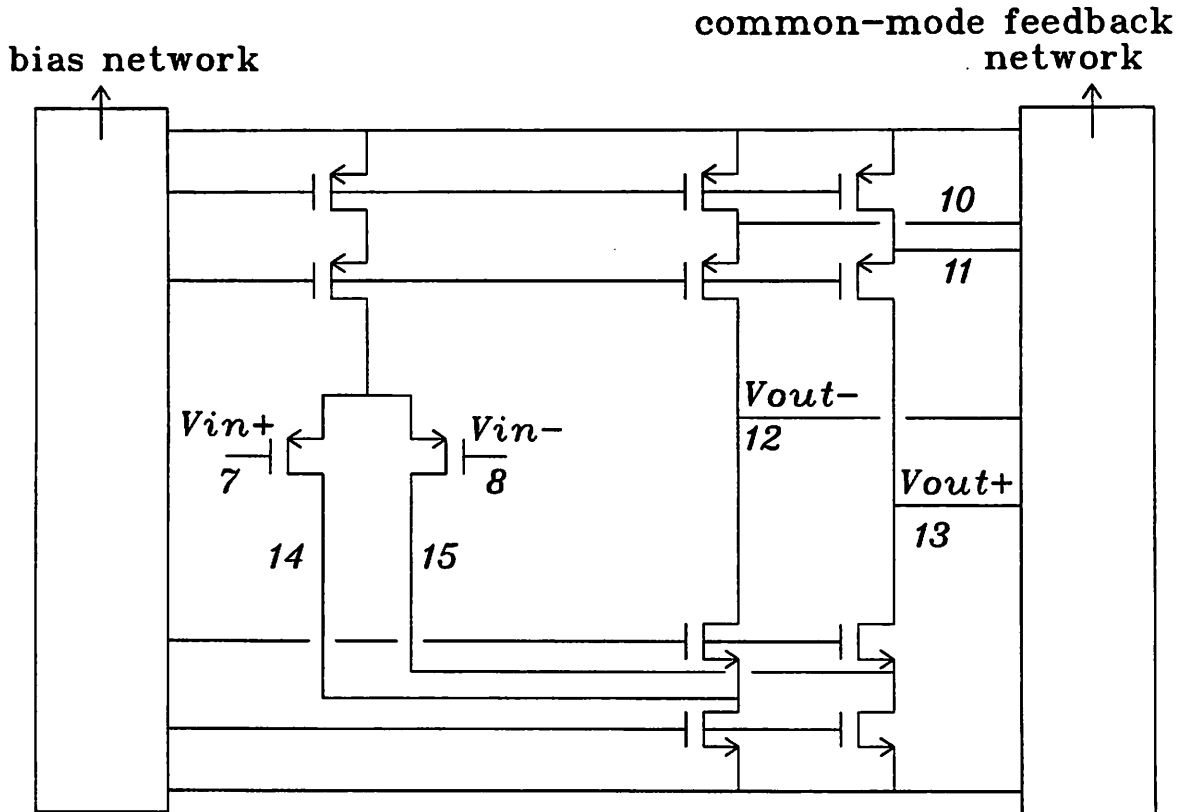


Figure 4.6: Fully-differential CMOS opamp.

shown in Fig. 4.6. SPICE3 was used for simulation and sensitivity computation. The performance functions of this opamp considered are (a) unity-gain bandwidth ( $ugb$ ), (b) phase margin ( $pm$ ) and (c) closed-loop gain ( $clg$ ). Performance functions  $ugb$  and  $pm$  are considered for the open-loop configuration and  $clg$  for the configuration shown in Fig. 4.7 (this configuration is often used in A/D conversion). The circuit has nominal  $ugb$  of 13.4 MHz, and  $pm$  of 79.5 degrees. These two performance functions have one-sided constraints (in the negative direction) and are nonprecision specifications (as explained in Section 4.2.1). It was decided to keep  $\Delta ugb$  and  $\Delta pm$  smaller than 0.1 MHz and 0.5 degree respectively in the negative direction (less than about 1% of their nominal values). The closed-loop gain

of the circuit is however a precision specification having two sided performance constraints. The ideal value of gain  $W_{ideal}$  is 2, and the nominal value  $W_{nom}$  is 1.994 (without any interconnect parasitics), mainly because of the finite open-loop gain of the opamp. The process variations  $\Delta W_{proc}^+$  and  $\Delta W_{proc}^-$  are 0.002 (computed as suggested in Section 4.2.2 by considering a worst-case mismatch of 0.1% between capacitors  $C_1$  and  $C_2$ ). Hence, the nominal precision  $W_{prec}^0$  without any interconnect parasitics is  $(2-1.994) + 0.002 = 0.008$ . The maximum allowed degradation  $\Delta W_{prec}$  in precision due to the parasitics has been set to 0.0005. Hence, from (4.15) and (4.16),  $\Delta W_{max}^+ = 0.0165$  and  $\Delta W_{max}^- = 0.0005$  for the closed-loop gain.

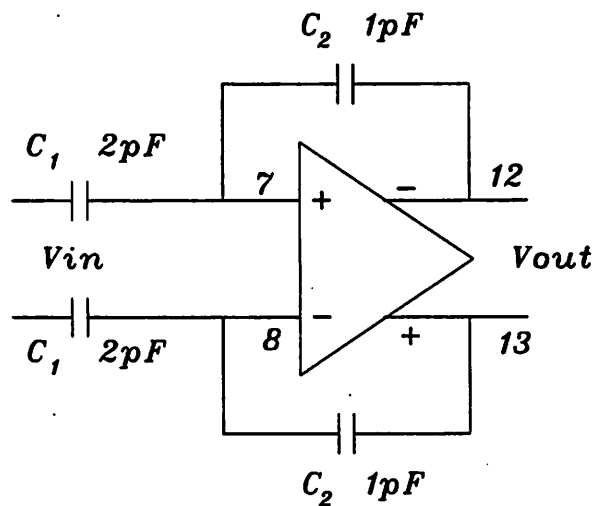


Figure 4.7: Closed-Loop Configuration.

The placement and global interconnect for the layout of the opamp has been done manually. The opamp has 37 nodes. The circuit has 7 matched pairs of nodes including the pairs (7,8), (10,11), (12,13) and (14,15) which are shown in Fig. 4.6. Matching Constraints were directly derived from matched-node-pair information as illustrated in Section 4.5. Sensitivity analysis was performed using SPICE3. The unity-gain bandwidth was found most sensitive to coupling capacitances involving nets 12 and/or 13, since they contribute the dominant poles. Nodes 14 and 15 on the other hand contribute the second dominant poles making the phase margin most sensitive to the coupling capacitances involving one or both of these nodes. The closed-loop gain as expected, is highly sensitive to coupling between input (7,8) and output (12,13) nodes of the opamp. The sensitivities of the performance functions with respect to some of the critical parasitics have been presented in Table 4,

Table 5, and Table 6. *net1* and *net2* refer to the two nets between which the parasitic capacitance may exist. Sensitivities with respect to matched parasitics have been computed taking the matching constraints into account. Bounding constraints were generated on 47 coupling capacitances. The opamp has been routed by the constraint-driven area router RoAD[93]. Five of the bounding constraints as well as the actual capacitances after routing are shown in Table 7. More detailed results regarding the constraint-driven area routing are provided in [93].

<i>net1</i>	<i>net2</i>	<i>ugb sensitivity</i>
12	13	5.5 kHz/fF
12	15	2.85 kHz/fF
10	13	2.75 kHz/fF
7	13	1.7 kHz/fF
12	0	1.4 kHz/fF

Table 4: Sensitivities of unity-gain-bandwidth w.r.t five of the most sensitive capacitances.

<i>net1</i>	<i>net2</i>	<i>pm sensitivity</i>
14	15	0.015 deg/fF
7	14	0.01 deg/fF
10	15	0.0075 deg/fF
7	10	0.0065 deg/fF
14	0	0.0038 deg/fF

Table 5: Sensitivities of phase margin w.r.t five of the most sensitive capacitances.

<i>net1</i>	<i>net2</i>	<i>clg sensitivity</i>
7	12	0.002/fF
7	13	-0.002/fF

Table 6: Sensitivities of closed-loop-gain w.r.t two of the most sensitive capacitances.



<i>net1</i>	<i>net2</i>	$C_{j\_bound}$	$C_{j\_actual}$
7	12	0.001fF	0fF
14	15	0.924fF	0.036fF
4	13	0.015fF	0.053fF
12	14	9.6fF	0.035fF
3	15	5.7fF	0.4fF

Table 7: Five of the bounding constraints on coupling capacitances and the actual estimated capacitances after routing.

The CPU time taken by the constraint generator to generate the constraints was about 25sec on a VAX 8650. Worst-case error due to linear approximations were found to be less than 2.0%, which means that the total performance degradation caused (when the parasitics are below their bounding constraints) may exceed specified values by 2%. But, since only conservative values need to be specified for  $\Delta W_{max}^+$  and  $\Delta W_{max}^-$ , these errors are not critical unless they are too large.

## 4.9 Summary

Algorithms for generating parasitic constraints from the specified performance constraints, were presented in this chapter. Linear approximations using sensitivities were used to model the performance constraints. For the test examples illustrated in this chapter, it was noted that these approximations are acceptable. Moreover, since standard circuit simulators have sensitivity computation capabilities, this approach is efficient from an implementation point of view. Worst case modeling was used considering process variations. The problem of distributing the parasitic constraints for maximizing the flexibility for the layout tools has been formulated as an optimization problem with quadratic cost function with linear performance constraints. A parasitic constraint generator PARCAR which implements the constraint-generation algorithms was reported. The output of the constraint generator can be used to drive the layout tools. However, the constraint generator can also be a valuable design aid, in its own right as it provides useful insight and guidelines to layout designers.

## 4.10 Appendix: Worst-Case Sensitivity with Process Variations

The expressions for upper and lower bounds for sensitivity with respect to  $p_0$  (value of  $p_j$  and  $p_k$  without any process variations) are now derived taking process variations into account. Let  $r_{max}$  be the maximum fractional process variation associated with a parasitic. Then, the actual values of the parasitics in terms of  $p_0$

$$p_j = (1 + r_j)p_0, \quad p_k = (1 + r_k)p_0 \quad (4.55)$$

where

$$|r_j|, |r_k| \leq r_{max} \quad (4.56)$$

If the values of the two parasitics were statistically independent of each other, then the maximum possible mismatch between the parasitics would have been  $2r_{max}$ . But, the actual mismatch will be less than  $2r_{max}$ , since the two parasitics will tend to track each other at least to some extent on the same chip.

$$|r_j - r_k| \leq r_{mis} \quad (4.57)$$

where

$$r_{mis} \leq 2r_{max} \quad (4.58)$$

Both  $r_{mis}$  and  $r_{max}$  are positive numbers.

Differentiating (4.55), as in (4.26), the sensitivity with process variation becomes

$$S = (1 + r_j)S_{ij} + (1 + r_k)S_{ik} \quad (4.59)$$

$$= S_0 + r_j S_{ij} + r_k S_{ik} \quad (4.60)$$

where  $S_0$  is given by (4.27). The goal is to compute the upper and lower bounds on the last two terms of (4.60) (contributions of process variation). This is a linear programming problem in  $r_j$  and  $r_k$  with a cost function (denoted by *cost*)

$$r_j S_{ij} + r_k S_{ik} \quad (4.61)$$

with (4.56) and (4.57) as the linear constraints.

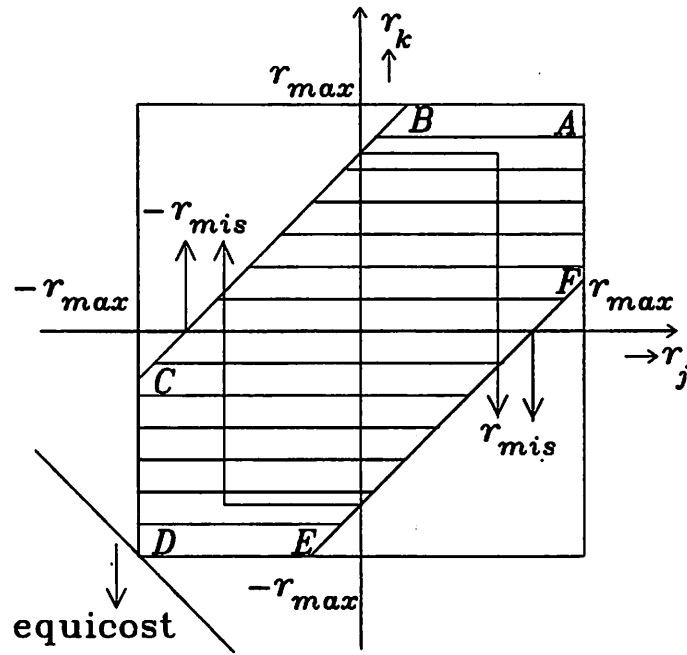


Figure 4.8: Sensitivity with process variations.

The problem has been illustrated with the aid of Fig. 4.8. The linear inequalities represented by (4.56) define a square box of dimension  $r_{max}$  in the two-dimensional space described by  $r_j$  and  $r_k$ . The additional mismatch inequalities represented by (4.57) cut off the shaded region of the square as shown, the two inclined lines (BC and EF) having slopes of 1. The cost function defined by (4.61) is constant on a straight line (referred to as *equicost* in Fig. 4.8) with a slope of  $-S_{ij}/S_{ik}$ . Note that, (4.58) always restricts the points B,C,E and F on the respective edges of the bounding square, since the points at which BC and EF meet the axes do not go beyond the range  $(2r_{max}, -2r_{max})$ .

**CASE I:**  $S_{ij}$  and  $S_{ik}$  are of same sign.

**case (Ia):**  $S_{ij}, S_{ik} \geq 0$

$$-r_{max} \leq r_j, r_k \leq r_{max} \quad (4.62)$$

$$\Rightarrow -r_{max}S_{ij} \leq r_jS_{ij} \leq r_{max}S_{ij} \quad (4.63)$$

$$-r_{max}S_{ik} \leq r_kS_{ik} \leq r_{max}S_{ik} \quad (4.64)$$

$$\Rightarrow -r_{max}(S_{ij} + S_{ik}) \leq r_jS_{ij} + r_kS_{ik} \leq r_{max}(S_{ij} + S_{ik}) \quad (4.65)$$

Thus,  $r_j S_{ij} + r_k S_{ik}$  attains its maximum and minimum at  $(r_{max}, r_{max})$  and  $(-r_{max}, -r_{max})$  respectively (which correspond to points  $A$  and  $D$  in Fig. 4.8 respectively).

case (Ib):  $S_{ij}, S_{ik} < 0$

Since  $S_{ij}$  and  $S_{ik}$  are negative, the inequalities in (4.63), (4.64) and (4.65) now go in opposite directions. The maximum and minimum in cost function are achieved at  $(-r_{max}, -r_{max})$  and  $(r_{max}, r_{max})$  respectively.

Hence, when  $S_{ij}$  and  $S_{ik}$  are of the same sign, the two limits for the range of sensitivity  $S$  are given by  $(1 + r_{max})S_0$  and  $(1 - r_{max})S_0$ . The maximum mismatch of parasitics does not come into the picture in determining the range of the sensitivity.

CASE II:  $S_{ij}$  and  $S_{ik}$  are of opposite sign.

case (IIa):  $|S_{ij}| = |S_{ik}|$

Since  $S_{ik} = -S_{ij}$ ,  $cost = S_{ij}(r_j - r_k)$ . But  $|r_j - r_k| \leq r_{mis}$  implies that upper and lower limits of  $cost$  are  $|S_{ij}|r_{mis}$  and  $-|S_{ij}|r_{mis}$  attained at  $|r_j - r_k| = \pm r_{mis}$ , which correspond to points on  $EF$  and  $BC$  in Fig. 4.8 respectively. Although the expressions for this case can be derived by setting  $|S_{ij}| = |S_{ik}|$  in the expressions for case (Iib) and case (Iic), this case has been treated separately since both the maximum and minimum occur over a continuous set of points in the  $(r_j, r_k)$  space rather than single points as in the other two cases.

case (Iib):  $|S_{ij}| < |S_{ik}|$

**Claim:**  $cost$  attains its extreme values at  $(r_{max} - r_{mis}, r_{max})$  and  $(r_{mis} - r_{max}, -r_{max})$  (points  $B$  and  $E$  respectively in Fig. 4.8).

**Proof:** At point  $B$ ,

$$cost(B) = (r_{max} - r_{mis})S_{ij} + r_{max}S_{ik} \quad (4.66)$$

$$= \pm[(r_{max} - r_{mis})|S_{ij}| - r_{max}|S_{ik}|] \quad (4.67)$$

'+' sign when  $S_{ij} \geq 0$  and  $S_{ik} < 0$ ; '-' sign when  $S_{ij} < 0$  and  $S_{ik} \geq 0$ .

$$cost(B) = \pm[r_{max}(|S_{ij}| - |S_{ik}|) - r_{mis}|S_{ij}|] \quad (4.68)$$

$$= \mp[r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}|] \quad (4.69)$$

$$= \mp C_{max} \quad (4.70)$$

where  $C_{max} = r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| > 0$ , since  $|S_{ik}| > |S_{ij}|$ .

Similarly, at point  $E$

$$cost(E) = (r_{mis} - r_{max})S_{ij} - r_{max}S_{ik} \quad (4.71)$$

$$= \pm[(r_{mis} - r_{max})|S_{ij}| + r_{max}|S_{ik}|] \quad (4.72)$$

$$= \pm[r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}|] \quad (4.73)$$

$$= \pm C_{max} \quad (4.74)$$

Now it will be shown that

$$-C_{max} \leq r_j S_{ij} + r_k S_{ik} \leq C_{max} \quad (4.75)$$

Let  $diff^+ = C_{max} - r_j S_{ij} - r_k S_{ik}$ . Substituting expression for  $C_{max}$ ,

$$diff^+ = r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| - r_j S_{ij} - r_k S_{ik} \quad (4.76)$$

$$= r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| - (r_j - r_k + r_k)S_{ij} - r_k S_{ik} \quad (4.77)$$

$$= r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| - (r_j - r_k)S_{ij} - r_k(S_{ij} + S_{ik}) \quad (4.78)$$

If  $S_{ij} \geq 0$  and  $S_{ik} < 0$ , then  $S_{ij} = |S_{ij}|$ ,  $S_{ik} = -|S_{ik}|$ . Thus,

$$diff^+ = r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| - (r_j - r_k)|S_{ij}| + r_k(|S_{ik}| - |S_{ij}|) \quad (4.79)$$

$$= (r_{max} + r_k)(|S_{ik}| - |S_{ij}|) + [r_{mis} - (r_j - r_k)]|S_{ij}| \quad (4.80)$$

The first term in the RHS of the above equation is nonnegative since  $|S_{ik}| > |S_{ij}|$  and  $|r_k| \leq r_{max}$ . The second term is nonnegative since  $|r_j - r_k| \leq r_{mis}$ . Hence,  $diff^+ \geq 0$ .

If  $S_{ij} < 0$  and  $S_{ik} \geq 0$ , then  $S_{ij} = -|S_{ij}|$ ,  $S_{ik} = |S_{ik}|$ .

$$diff^+ = (r_{max} - r_k)(|S_{ik}| - |S_{ij}|) + [r_{mis} + (r_j - r_k)]|S_{ij}| \quad (4.81)$$

Again,  $dif^+ \geq 0$  due to same reasons.

$$dif^+ \geq 0 \Rightarrow r_j S_{ij} + r_k S_{ik} \leq C_{max} \quad (4.82)$$

Let  $dif^- = r_j S_{ij} + r_k S_{ik} - (-C_{max})$ . Substituting expression for  $C_{max}$ ,

$$dif^- = r_j S_{ij} + r_k S_{ik} + r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| \quad (4.83)$$

$$= (r_j - r_k + r_k)S_{ij} + r_k S_{ik} + r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| \quad (4.84)$$

$$= (r_j - r_k)S_{ij} + r_k(S_{ij} + S_{ik}) + r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}| \quad (4.85)$$

If  $S_{ij} \geq 0$  and  $S_{ik} < 0$  ( $S_{ij} = |S_{ij}|$ ,  $S_{ik} = -|S_{ik}|$ ),

$$dif^- = (r_{max} - r_k)(|S_{ik}| - |S_{ij}|) + [r_{mis} + (r_j - r_k)]|S_{ij}| \quad (4.86)$$

Arguing as before for  $dif^+$ ,  $dif^- \geq 0$ .

If  $S_{ij} < 0$  and  $S_{ik} \geq 0$  ( $S_{ij} = -|S_{ij}|$ ,  $S_{ik} = |S_{ik}|$ ),

$$dif^- = (r_{max} + r_k)(|S_{ik}| - |S_{ij}|) + [r_{mis} - (r_j - r_k)]|S_{ij}| \quad (4.87)$$

Again,  $dif^- \geq 0$ .

$$dif^- \geq 0 \Rightarrow r_j S_{ij} + r_k S_{ik} \geq -C_{max} \quad (4.88)$$

From (4.82) and (4.88), it is concluded that the upper and lower bounds of *cost* are  $\pm C_{max}$  i.e.  $\pm[r_{max}(|S_{ik}| - |S_{ij}|) + r_{mis}|S_{ij}|]$  respectively.

**case (IIc):**  $|S_{ij}| > |S_{ik}|$

**Claim:** *cost* attains its extreme values at  $(r_{max}, r_{max} - r_{mis})$  and  $(-r_{max}, r_{mis} - r_{max})$  (points *F* and *C* respectively in Fig. 4.8).

**Proof:** Proof is identically same as for case (IIb) with the roles of subscripts *j* and *k* interchanged in  $S_{ij}$ ,  $S_{ik}$ ,  $r_j$  and  $r_k$ . The upper and lower bounds of *cost* are

$\pm[r_{max}(|S_{ij}| - |S_{ik}|) + r_{mis}|S_{ik}|]$  respectively.

One has to add  $S_0$  to the bounds for *cost* to get the bounds for the sensitivity (refer to (4.60)).

The need for taking process variations into account while computing sensitivities will now be illustrated with a practical example. Consider a differential amplifier circuit having two input nodes which are extremely sensitive to external noise. If a large swing net comes close to these two nodes, there will be a matching constraint imposed on the coupling capacitances as in (4.34). Without any process variations, the sensitivities of the amplifier output to the two coupling capacitances will be equal in magnitude and opposite in sign i.e. the net sensitivity is zero if computed using (4.27). This will result in *no bounding constraint* on the two capacitances. But in practice, the two coupling capacitances may have a mismatch of as large as 20%, since the routing capacitances are poorly controlled. As each of the capacitances gets bigger (when the large-swing net comes closer), the absolute value of the maximum mismatch also grows proportionately, increasing the worst-case noise voltage at the amplifier output (since the two sensitivities act in opposite direction). This gives rise to the need for bounding constraints on these coupling capacitances.

When sensitivity has to be computed with respect to a single parasitic which is not to be matched with any other parasitic, then proceeding in similar fashions, it can be shown that the two bounds for the actual sensitivity  $S$  are  $(1 + r_{max})S_0$  and  $(1 - r_{max})S_0$ , where  $S_0$  is the sensitivity computed without taking into account any process variations. Hence  $(1 + r_{max})S_0$  can be used as the worst-case sensitivity.

## 4.11 Appendix: Estimation of Maximum and Minimum Capacitances

A-priori estimates of maximum and minimum values for parasitics are used during constraint generation (Section 4.6). The estimation algorithms presented here apply for the channel-routing algorithm used in the router ART, for a given placement and global routing. It is assumed that there is a single horizontal segment for each net at a variable position, and a vertical segment at the x-position of each fixed or floating pin on top and

bottom edge. To make the algorithms transparent to the reader, net-width and channel-to-net spacing have been assumed to be zero, although the actual algorithms take those into account. For the same reason, the scanning techniques used by the program to scan the data structures in a sorted order are also not shown explicitly in the algorithms. For each main channel, the horizontal direction is taken along the x-axis and the vertical one along the y-axis.

To make an a-priori estimate of upper and lower limits for various interconnect capacitances in the layout, all the channels have to be considered before routing begins. When none of the channels is routed, it is possible to have some floating pins on the horizontal (top and bottom) edges of a given channel (referred to as the main channel), which correspond to floating nets associated with the adjacent channels, oriented at a right angle to the main channel as shown in Fig. 4.9. These adjacent top and bottom channels have to be routed before the main channel.

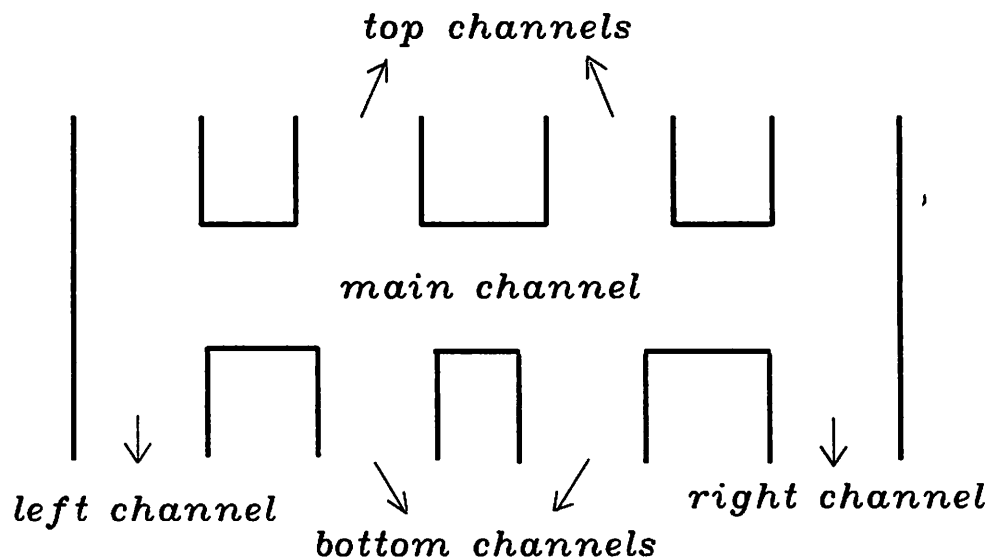


Figure 4.9: Adjacent channels of a channel.

For a given channel, total coupling between two nets can contain contributions from coupling between adjacent *parallel horizontal segments*, *parallel vertical segments* and *crossing segments* inside the channel. Moreover, there can be coupling between two parallel lines, one inside the channel, and the other in an adjacent channel (which can be on the left, right, top or bottom of the main channel). To avoid considering these *inter-channel couplings* twice, the coupling between the main channel and only the top and bottom



channels is taken into account while processing a given main channel. This choice is made, since the top and bottom channels are routed before the main channel. Hence, it is possible to estimate the associated couplings to meet the bounding constraint (if any), while routing the main channel. Thus, one can take care of all possible inter-channel couplings when the channels are routed in the appropriate order.

#### 4.11.1 Horizontal Segments

The minimum capacitance between two horizontal segments is assumed to be zero (when the segments move apart, and other nets go in between). Hence, only the maximum possible capacitance between two horizontal segments is estimated. Due to the presence of floating pins on the top and bottom edges of the main channel, the horizontal span of any net is not fixed before routing.

The following algorithm computes the left and right positions of maximum and minimum spans of a net (denoted by *max\_span\_left*, *max\_span\_right*, *min\_span\_left* and *min\_span\_right* respectively). Floating pins on top/bottom edges are assumed to be on the leftmost positions of the corresponding top/bottom channels, for estimating *max\_span\_left* and *min\_span\_right*, and on the rightmost positions for estimating *max\_span\_right* and *min\_span\_left*. LEFT() and RIGHT() functions give the x positions of left and right edges of a channel.

**Algorithm 3 (Estimation of left and right positions of maximum and minimum horizontal spans of a net)**

```
(estimation of max_span_left and min_span_left)
if (net crosses left edge of the main channel) then
    max_span_left = min_span_left = LEFT(main channel);
else begin
    max_left = min_left = x position of leftmost fixed pin;
    for each top channel do begin
        if (net crosses bottom of the top channel) then begin
            max_left = MIN(max_left,LEFT(top channel));
            min_left = MIN(min_left,RIGHT(top channel));
```

```

    end
  end
  for each bottom channel do begin
    if (net crosses top of the bottom channel) then begin
      max_left = MIN(max_left,LEFT(bottom channel));
      min_left = MIN(min_left,RIGHT(bottom channel));
    end
  end
  end
  max_span_left = max_left; min_span_left = min_left;
end

```

(estimation of *max\_span\_right* and *min\_span\_right*)

change left to right, and MIN to MAX in the above algorithm;

Let *max\_span\_left* and *max\_span\_right* of *net1* and *net2* denote *left1*, *right1*, *left2* and *right2* respectively. The maximum and minimum common horizontal spans of the two nets denoted by *max\_common\_span* and *min\_common\_span* can be computed by taking the intersections of corresponding span intervals obtained for each net from Algorithm 3. An illustrative example is depicted in Fig. 4.10, where the *net1* vertical segment shown by the broken line can be present anywhere in the span of the top channel. This has to be taken into account to calculate the maximum and minimum common spans. Only, the *max\_common\_span* of two nets is used in this section, while both *max\_common\_span* and *min\_common\_span* are used for estimations involving crossing segments.

The function INTERSECT() gives the length of the intersection of two intervals. *tech\_dmin* is the minimum distance between two adjacent lines allowed by the technology. CAP\_ADJACENT() is the model which computes the per-unit-length capacitance between two adjacent parallel lines.

**Algorithm 4 (Estimation of maximum coupling capacitance between horizontal segments of net1 and net2)**

(coupling inside the channel)

```

max_common_span =
    INTERSECT(( left1, right1 ),( left2, right2 ));
lmax = max_common_span;
dmin = tech.dmin ;
Cmax = lmax * CAP_ADJACENT(dmin);

(interchannel coupling)
(net1 inside and net2 outside)
for each top/bottom channel do begin
    dmin = distance of closest pin of net2
        from top/bottom edge of main channel;
    if (dmin < ID ) do begin
        leftc = LEFT(top/bottom channel);
        rightc = RIGHT(top/bottom channel);
        lmax = INTERSECT((left1,right1),(leftc,rightc));
        Cmax = Cmax + lmax * CAP_ADJACENT(dmin);
    end
(net2 inside and net1 outside)
    swap net1 and net2 in the above algorithm;

```

#### 4.11.2 Vertical Segments

Coupling occurring between vertical segments only *inside* the channel are considered, since as mentioned earlier, coupling with respect to segments in left and right channels are handled while processing those channels to avoid duplication. The x-position of each floating pin on the top or bottom edge is treated as a variable in the range spanned by the corresponding top/bottom channel for estimation of maximum possible capacitances. The algorithm presented below considers all possible pairs of *net1* and *net2* vertical segments which can run adjacent to each other, and estimates the minimum possible distance between the vertical segments in each case. If this distance is less than *ID*, the maximum estimate of capacitance is incremented by the appropriate amount. The maximum common span between two vertical segments is taken as the channel width. The minimum capacitance is

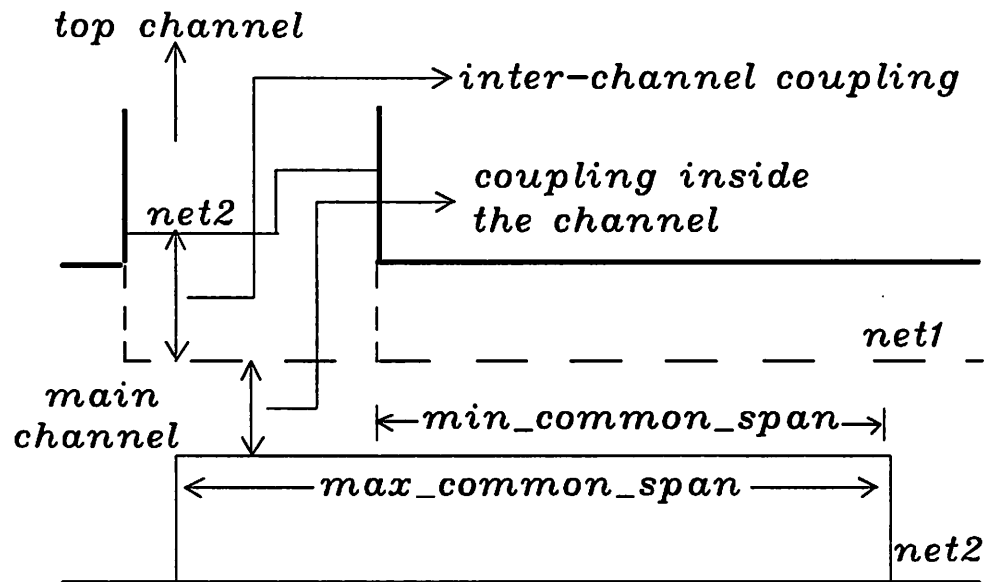


Figure 4.10: Coupling between horizontal segments.

again taken to be zero as for horizontal segments. As mentioned earlier, a vertical segment is associated with each fixed and floating pin on top or bottom edge of the main channel. An example is shown in Fig. 4.11.

To satisfy design rules, vertical segments of pins lying within a distance of  $tech\_dmin$  on opposite edges cannot run adjacent to each other. For the same reason, two pins on the same edge cannot lie within a distance of  $tech\_dmin$ . This is assumed in the following algorithm. Hence, at any stage, if the minimum possible distance between two vertical segments is estimated as  $tech\_dmin$ , there is no need to do further computation for that pair, although that is not explicitly shown here for conciseness.

**Algorithm 5 (Estimation of maximum coupling capacitance between vertical segments of net1 and net2)**

(net1 pin fixed, net2 pin fixed or floating )

for each fixed net1 pin do begin

net1\_x = x position of the net1 pin;

net1\_left = net1\_x -  $tech\_dmin$  ;

net1\_right = net1\_x +  $tech\_dmin$  ;

(Now successively consider fixed and floating pins of net2 on the same edge as net1 pin, and then on the opposite edge)

**(net2 pins on the same edge)**

(all the following distances are measured from net1\_x)

d\_left(or d\_right) = smaller of the distances of closest net2

fixed pin, and closest possible net2 floating pin to the

left(or right) of net1\_left(or net1\_right) on the same edge;

**(net2 pins on the opposite edge)**

(fixed net2 pins)

opp\_left(or opp\_right) = distance of the closest net2 fixed pin

to the left(or right) of net1\_left(or net1\_right) on the

opposite edge;

d\_left = MIN(d\_left,opp\_left);

d\_right = MIN(d\_right,opp\_right);

(floating net2 pins)

for each top/bottom channel on the opposite edge do begin

(the top/bottom channel is denoted as opchan)

if (net2 crosses common edge of opchan and

the main channel) then begin

if (RIGHT(opchan)  $\leq$  net1\_left) then

d\_left = MIN(d\_left,net1\_x-RIGHT(opchan));

else if (LEFT(opchan)  $\geq$  net1\_right) then

d\_right = MIN(d\_right,LEFT(opchan)-net1\_x);

else d\_left = d\_right = *tech\_dmin* ;

end

end

end

**(net1 pin floating, net2 pin fixed or floating )**

for each floating pin of net1 do begin

(channel containing the net1 floating pin is denoted

as net1\_chan)

```

net1_left = LEFT(net1_chan) - tech_dmin ;
net1_right = RIGHT(net1_chan) + tech_dmin ;

(net2 pins on the same edge)
(the following distances are measured from LEFT(net1_chan)
for d_left, and from RIGHT(net1_chan) for d_right)
d_left(or d_right) = smaller of the distances of closest net2
    fixed pin, and closest possible net2 floating pin to the
    left(or right) of net1_left(or net1_right) on the same edge;

(net2 pins on the opposite edge)
(fixed net2 pins)
for each fixed pin of net2 on opposite edge do begin
    net2_x = x position of the net2 pin;
    if (net2_x ≤ net1_left) then
        d_left = MIN(d_left, net1_left - net2_x);
    else if (net2_x ≥ net1_right) then
        d_right = MIN(d_right, net2_x - net1_right);
    else d_left = d_right = tech_dmin ;
end

(floating net2 pins)
for each top/bottom channel on the opposite edge do begin
    (the top/bottom channel is denoted as opchan)
    if (net2 crosses common edge of opchan and
        the main channel) then begin
        if (RIGHT(opchan) ≤ net1_left) then
            d_left =
                MIN(d_left, LEFT(net1_chan) - RIGHT(opchan));
        else if (LEFT(opchan) ≥ net1_right) then
            d_right =
                MIN(d_right, LEFT(opchan) - RIGHT(net1_chan));
        else d_left = d_right = tech_dmin ;
    end
end
end

```

end

**(final estimates)**

$d\_min\_left = d\_left; d\_min\_right = d\_right;$

$lmax = \text{width of the main channel};$

$Cmax = 0;$

if( $d\_min\_left < ID$ ) then

$Cmax = Cmax + lmax * CAP\_ADJACENT(d\_min\_left);$

if( $d\_min\_right < ID$ ) then

$Cmax = Cmax + lmax * CAP\_ADJACENT(d\_min\_right);$

During estimation of maximum and minimum capacitances, to check the validity of value of influence distance  $ID$  used in the model, one can mark each situation in which coupling between a possible pair of *adjacent* parallel segments is ignored because they are going to run at a distance larger than  $ID$  (refer to Algorithms 4 and 5). Then the worst-case contribution of all those couplings which are ignored can be computed by making the separation in each marked case equal to  $ID$ , and should be compared with the maximum performance degradation allowed.

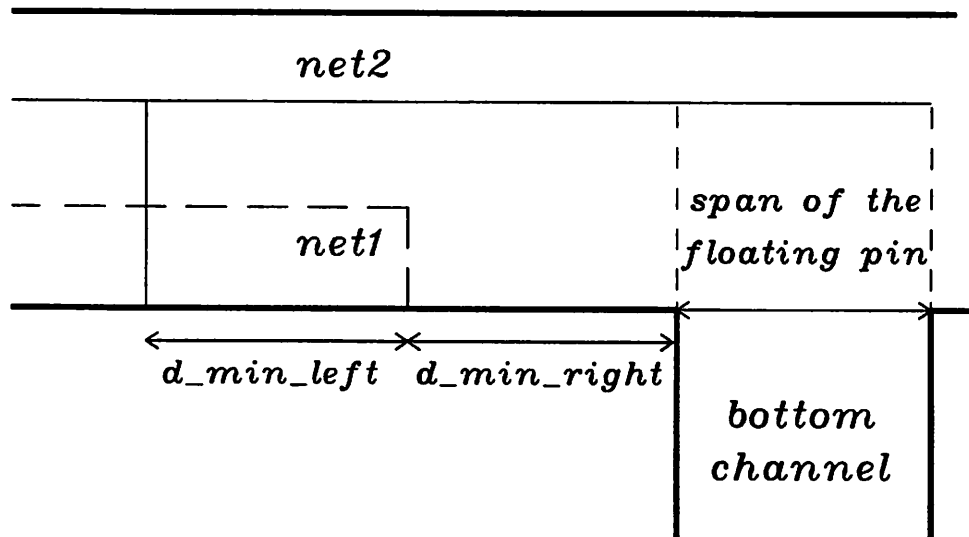


Figure 4.11: Coupling between vertical segments.

### 4.11.3 Crossing Segments

A crossing takes place inside the channel, between a horizontal segment of *net1* and a vertical segment of *net2*, when a pin of *net2* (say  $p_2$ ) is inside the common horizontal span of the two nets, and one of the following conditions is satisfied.

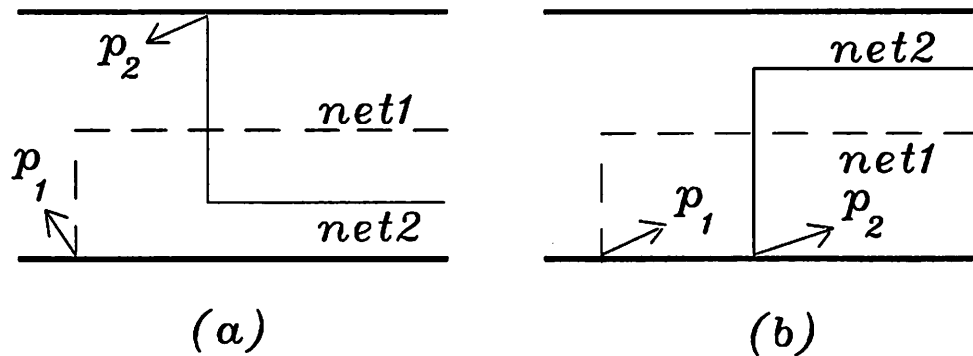


Figure 4.12: Coupling between crossing segments.

(a) pin  $p_2$  is on the top edge, and the horizontal segment of *net2* is below that of *net1* as shown in Fig 4.12(a).

(b) pin  $p_2$  is on the bottom edge, and the horizontal segment of *net2* is above that of *net1* as shown in Fig 4.12(b).

Hence, for estimating maximum and minimum crossing capacitances, one needs to know the maximum and minimum common horizontal spans of *net1* and *net2*. The intervals ( $max\_span\_left$ ,  $max\_span\_right$ ) and ( $min\_span\_left$ ,  $min\_span\_right$ ) which are computed by Algorithm 3 are denoted by  $max\_common\_range$  and  $min\_common\_range$  respectively.

There are two different cases regarding the relative positions of horizontal segments of *net1* and *net2*, and the associated maximum and minimum crossover capacitances are denoted respectively by the quantities  $max\_cross\_case1$ ,  $max\_cross\_case2$ ,  $min\_cross\_case1$  and  $min\_cross\_case2$ . CROSS\_INCREMENT() is a subroutine which increments its argument by the crossover capacitance for each crossover detected, using the capacitance model. The estimation algorithm using these notations is given below.



**Algorithm 6 (Estimation of maximum and minimum crossover capacitance between net1 and net2)**

**(case 1: net1 horizontal segment above that of net2)**

*max\_cross\_case1* = *min\_cross\_case1* = 0;

**(top pins of net2)**

for each top pin of net2 do begin

  if (net2 pin is fixed) then begin

    if (x position of net2 pin belongs to

*max\_common\_range*) then

      CROSS.INCREMENT(*max\_cross\_case1*);

    if (x position of net2 pin belongs to

*min\_common\_range*) then

      CROSS.INCREMENT(*min\_cross\_case1*);

  end

  else begin

    (check if crossing *may* occur):

    if (span of top channel containing floating net2 pin

      intersects *max\_common\_range*) then

      CROSS.INCREMENT(*max\_cross\_case1*);

    (check if crossing is *bound* to occur):

    if (span of top channel containing floating net2 pin

      is contained in *min\_common\_range*) then

      CROSS.INCREMENT(*min\_cross\_case1*);

  end

end

**(bottom pins of net1)**

  replace net2 by net1 and top by bottom in the above algorithm;

**(case 2: net2 horizontal segment above that of net1)**

  swap net1 and net2 in the algorithm for case1; .

(final estimates)

$$C_{\max} = \text{MAX}(max\_cross\_case1, max\_cross\_case2);$$

$$C_{\min} = \text{MIN}(min\_cross\_case1, min\_cross\_case2);$$

The total maximum and minimum estimates of coupling capacitance between two nets is obtained by adding over all the channels the corresponding contributions from horizontal, vertical and crossing segments.

## 4.12 Appendix: Shielding Decision

It is easy to see that there can be a nonzero value of minimum crossover capacitance decided by pin positions of the nets (for example, when pins of a net are present both on top and bottom edges of a channel, and another net has to go right through the channel from the left to the right edge). For a given placement and global routing, the minimum crossover capacitance for the channel routing algorithm used in ART (which is interfaced to the parasitic constraint generator PARCAR) can be estimated as described in Appendix 4.11. It was mentioned earlier (Section 4.6.4) that the minimum values of parasitics should form a feasible solution for a successful generation of bounding constraints. If that is not the case, some of the crossovers have to be *shielded*, to reduce the minimum possible capacitances. Shielding can be done when a third layer of interconnection is available (like POLY in addition to MET1 and MET2), which need not be used frequently for routing (since it is a high-resistivity material), but can be used for running short stretches of interconnects if necessary. In that case, the interconnect in one of the layers can be shifted to the third layer for a short length, and use a grounded plate in the intermediate layer to shield the crossover (for more detailed description of crossover shielding, refer to [16]). Shielding should be done only when it is not possible to meet performance constraints without it, since shielding requires additional ground wires to be brought to the shielding site, and hence increases the area occupied by the chip. Also, it increases the capacitive loading of neighboring lines.

Shielding decisions are made before constraint generation. To cause minimum possible shielding, for each performance function for which equation (4.50) or (4.51) is not satisfied by the minimum capacitances, the capacitances are sorted in decreasing magnitudes

of performance sensitivities, and eliminated successively (marked as shielded) till equations (4.50) and (4.51) are satisfied. Once a feasible solution is assured, the bounding constraints can be generated. The shielding decisions can be passed to the router along with the parasitic constraints. However, the router is free to go for additional shielding if necessary. During routing, it is also possible to run shielding wires between two parallel adjacent lines.

## Chapter 5

# Constraint-driven layout design

In the previous chapter, the problem of generating parasitic constraints from the performance constraints was addressed. In the performance-constrained approach, after this parasitic constraint generation, constraint-driven layout design has to be performed in which the layout tools are driven by the parasitic constraints. This phase contains constraint-driven placement and routing. The parasitics are actually determined during routing. The placement however imposes limits on the constraints which can be met during routing. For example the manhattan distances between pins impose lower bounds on the net resistances and capacitances. Hence, during placement the goal should be to make sure that routing is not overconstrained. In this chapter, the primary emphasis is on the problem of channel routing (the chosen form of channel routing is gridless) driven by parasitic constraints. The author has worked with other colleagues on constraint-driven area routing and placement. Hence only overviews of these contributions will be presented in this chapter with pointers to additional references.

Section 5.1 lists the notations used in this chapter. Section 5.2 contains an overview of gridless channel routing. Sections 5.3 and 5.4 respectively describe the algorithms for imposing bounding and matching constraints associated with the parasitics on channel routing. Section 5.5 describes the constraint-driven channel router ART. Section 5.6 results of constraint-driven channel routing for some test examples. Sections 5.7 and 5.8 contain overviews of constraint-driven area routing and placement. Section 5.9 summarizes this chapter.

## 5.1 Notations

$A, \bar{A}$ : matched pair of nets

$B, \bar{B}$ : matched pair of nets

$C_{bound}(i, j)$ : bound on capacitance between nets  $n_i$  and  $n_j$

$C_{c12}$ : capacitance of each crossover due to the horizontal segment of  $n_1$  and a vertical segment of  $n_2$

$C_{c21}$ : capacitance of each crossover due to the horizontal segment of  $n_2$  and a vertical segment of  $n_1$

$C_{cross12}$ : crossover capacitance between  $n_1$  and  $n_2$  when the horizontal segment of  $n_1$  is above that of  $n_2$

$C_{cross21}$ : crossover capacitance between  $n_1$  and  $n_2$  when the horizontal segment of  $n_2$  is above that of  $n_1$

$C_{cross\_max}$ : estimate of maximum crossover capacitance between  $n_1$  and  $n_2$

$C_{cross\_min}$ : estimate of minimum crossover capacitance between  $n_1$  and  $n_2$

$C_{h\_bound}$ : bound on capacitance between horizontal segments of  $n_1$  and  $n_2$

$C_h(d)$ : per-unit-length capacitance between adjacent horizontal segments at separation  $d$

$C_{h\_max}$ : estimate of maximum capacitance between horizontal segments of  $n_1$  and  $n_2$

$C(i, j)$ : capacitance between nets  $n_i$  and  $n_j$

$C_{max}$ : estimate of maximum capacitance between  $n_1$  and  $n_2$

$C_{min}$ : estimate of minimum capacitance between  $n_1$  and  $n_2$

$CP$ : set of critical pairs of nets

$C_v(d)$ : per-unit-length capacitance between adjacent vertical segments at separation  $d$

$C_{v\_max}$ : estimate of maximum capacitance between vertical segments of  $n_1$  and  $n_2$

$d_l$ : minimum center-line to center-line distance at which net  $B$  can be placed below merged net  $(A, \bar{A})$

$d_{low}(n_1, n_2)$ : estimate of lower bound on the minimum separation between horizontal segments of  $n_1$  and  $n_2$

$d_{min}$ : minimum edge-to-edge separation between two nets to satisfy design rules

$d_s$ : minimum center-line to center-line distance at which net  $B$  can be placed above merged net  $(A, \bar{A})$

$d_{sep}$ : minimum separation between horizontal segments necessary to meet the capacitance bound  $C_{h\_bound}$

$d_{vl}$ : minimum via-to-line separation (center-line to center-line) to satisfy design rules

$d_{vv}$ : minimum via-to-via separation (center-line to center-line) to satisfy design rules

$ID$ : influence distance (beyond which coupling is ignored)

$lucs(n_1, n_2)$ : length of the  $ucs(n_1, n_2)$

$LDP(Top, Bottom)$ : length of longest directed path from the source node to the sink node in the VC graph

$LDP(n_1, n_2)$ : length of longest directed path from node  $n_1$  to node  $n_2$  in the VC graph

$n$  : net

$n_s$  : shield net

$N_b(n_1), N_b(n_2)$ : number bottom pins of  $n_1$  and  $n_2$  respectively in the common horizontal span of  $n_1$  and  $n_2$

$N_c(n_1, n_2)$ : set of nets whose horizontal segments are constrained to lie between those of  $n_1$  and  $n_2$

$N_t(n_1), N_t(n_2)$ : number top pins of  $n_1$  and  $n_2$  respectively in the common horizontal span of  $n_1$  and  $n_2$

$N_s(n_1, n_2)$ : set of all shield nets whose horizontal spans intersect the common horizontal span of  $n_1$  and  $n_2$

$N_{sc}(n_1, n_2)$ : set of shield nets whose horizontal segments are constrained to lie between those of  $n_1$  and  $n_2$

$p$  : pin

$p_s$  : pin of a shield net

$P_{us}$ : set of pin pairs of  $n_1$  and  $n_2$  whose corresponding vertical segments are within influence distance and not shielded

$seg$  : horizontal segment

$seg_s$  : horizontal segment of a shield net

$span(n)$ : interval representing the horizontal span of a net  $n$

$S_{vc}(n_1, n_2)$ : set of VC graphs obtained by considering all feasible directions of the undirected edges between any two nodes representing nets in  $N_c(n_1, n_2)$

$ucs(n_1, n_2)$ : unshielded common span of  $n_1$  and  $n_2$

VC graph: Vertical-constraint graph

$w_A, w_{\bar{A}}$ : widths of nets  $A, \bar{A}$  respectively

$x_{lA}, x_{lB}, x_{l\bar{A}}$  : x coordinates of leftmost pins of nets  $A, B$  and  $\bar{A}$  respectively

$x_{rA}, x_{rB}, x_{r\bar{A}}$ , : x coordinates of rightmost pins of nets  $A, B$  and  $\bar{A}$  respectively

## 5.2 Overview of Gridless Channel Routing

As mentioned earlier, while using channel routing for complete layouts, the problem is broken into smaller problems of routing individual channels. A channel is usually a rectangular space. The routing assignment is usually divided into two subtasks (a) global routing and (b) detailed routing. During global routing, the rough paths of the nets (which span more than one channel) are determined by specifying the nets which have to cross the left and right edges of the channels. The exact positions of the nets in the channels are determined during detailed routing. In this section, an overview of detailed routing is presented. The rest of the paper addresses how parasitic constraints can be accounted for in the routing algorithm.

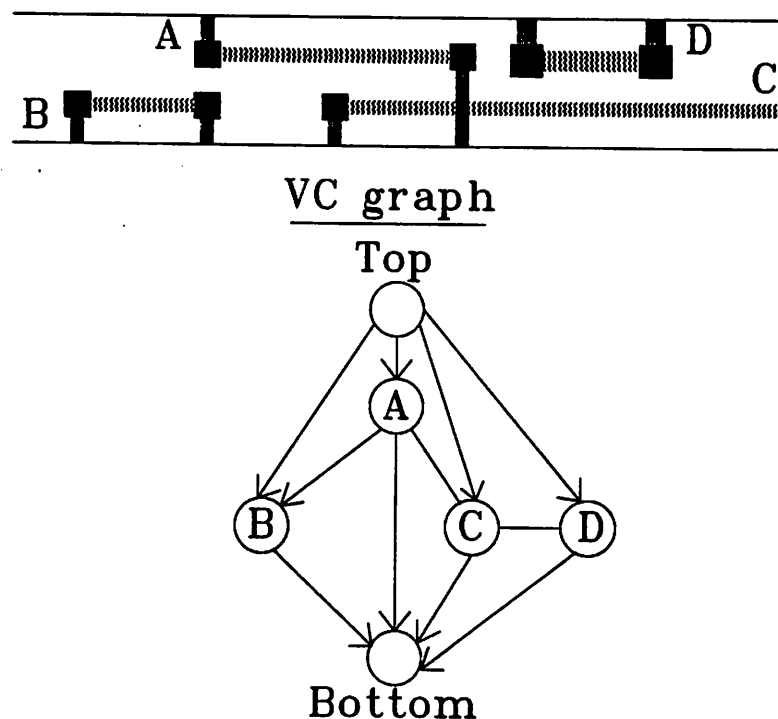


Figure 5.1: A channel and its VC graph

In gridless channel routing, the positions of pins and nets are not restricted to lie

on any grid. Hence, it is easy to handle variable-width nets and irregular top and bottom edges, features which are particularly useful for analog circuits.

For detailed routing in the channel, fixed pins are specified on the top and bottom edges of the channel and floating pins are specified on the left and right edges. If a floating pin is specified on an edge it implies that the net associated with that pin crosses that edge, but the actual position of the crossing will be determined during routing. In the gridless approach as described in [95], the channel-routing problem is represented by a *vertical-constraint graph*, which contains a node for each horizontal segment of a net(subnet). Horizontal and vertical segments are usually assigned to different layers. For simplicity, throughout this paper it is assumed that one horizontal segment is used to route each net, and one vertical segment connects the horizontal segment to each pin of the net (it may be necessary to remove this restriction to break cycles in the VC graph, and to reduce channel height <sup>1</sup>). Geometric constraints between horizontal segments of the nets in the channel are represented as edges in the VC graph. There is an undirected edge between two nodes if the associated segments have a common effective horizontal span (the effective horizontal span is the actual horizontal span with half of minimum net-to-net separation added to both sides), which represents the requirement that one segment has to be placed above or below the other segment at least at minimum allowed separation (e.g. nets *A* and *C* in Fig. 5.1), otherwise there will be a design rule violation since all the horizontal segments are in the same layer. There is a directed edge from a node of any net  $n_1$  to that of another net  $n_2$  if the horizontal segment of  $n_1$  has to be placed above that of  $n_2$  at least at minimum allowed spacing. This is necessary when a pin of  $n_1$  on top edge and a pin of  $n_2$  on bottom edge have common effective horizontal span, since otherwise there will be a design rule violation associated with the vertical segments connected to the pins. (e.g. nets *A* and *B* in Fig. 5.1).

The weight of an edge is the minimum distance between center lines of two adjacent horizontal segments allowed by design rules. The top and bottom edges of the channel correspond to the source and the sink nodes of the graph. There is a directed edge from the source node to the node associated with each net, and a directed edge from the node associated with each net to the sink node (since all the horizontal segments have to be placed below the top edge and above the bottom edge at least at the minimum allowed

---

<sup>1</sup>This can be achieved either by inserting horizontal jogs [94] or using a maze router as a post-processing step[82].



separation). The channel-routing problem is then formulated as directing the undirected edges to minimize the longest directed path in the graph from the source to the sink node. This physically means determining the relative positions of the horizontal segments of the nets to minimize the channel height. Once the relative positions of the segments are known the actual positions are determined based on minimum allowed net-to-net separations (represented by the weights of edges). The longest directed path from the source node to the sink node in the VC graph hence determines the channel height.

Since this problem is NP-complete, a heuristic algorithm is used to minimize the channel height as described in detail in [95]. A more general formulation can be developed for the problem when the top and bottom edges are irregular[95].

If the  $x$ -axis is assumed to be parallel to the top/bottom edges of the channel, then the *local density* at any  $x$ -position in the channel is the sum of the effective widths (actual width + minimum allowed net-to-net spacing) of all the horizontal segments passing through that position. The *channel density* is the maximum local density along the channel and gives a lower bound for the height of the channel. This lower bound is known before the routing is performed.

In Fig. 5.1, there is a directed edge from  $A$  to  $B$  because of the respective pin positions. There are two undirected edges ( $A,C$ ) and ( $C,D$ ). During channel routing, these two undirected edges have to be directed from  $A$  to  $C$ , and from  $D$  to  $C$  respectively to minimize the channel height.

Now the imposition of bounding and matching constraints associated with parasitics will be described.

### 5.3 Imposing the Bounding Constraints on Channel Routing

The parasitics which are controlled during routing are line resistances, line-to-ground capacitances and line-to-line capacitances (inductances can be important only at very high frequencies like microwaves). It is to be noted that the line resistances and line-to-ground capacitances of nets cannot be controlled to a significant extent during the detailed routing phase, once the widths and layers for horizontal and vertical segments of the nets are fixed. They can be bounded during placement and global routing. For analog

circuits, it is also true that the coupling capacitances are more important than the capacitances to ground. The routing capacitances to ground only add to the already existing device capacitances (usually larger in magnitude as devices tend to be big). However, even small coupling between certain pairs of nets can be quite detrimental for circuit performance as described in the introductory chapter. For example, in switched-capacitor circuits, unwanted capacitive coupling between interconnects can destroy ratio accuracy of precision capacitors. In layouts containing both digital and analog blocks, noise signal fed from digital to analog lines can impair the precision of analog circuits in a serious way, besides limiting their dynamic range.

If one of the two layers available for channel routing is a high-resistivity/capacity layer (such as POLY), and there is a requirement that a specific net has to be routed in only the low-resistivity/capacity layer, then it is possible to satisfy that constraint if that net does not cross any other net as noted in [28]. This can be achieved by imposing the crossover constraint between that net and all other nets. using the methodology described in Section 5.3.2. Hence, the constraints on capacitance to ground and resistance are not explicitly mentioned any more in this section, although they are considered in Section 5.4 while dealing with matching constraints. This section presents routing algorithms driven by a set of bounds on coupling capacitances between specific pairs of nets.

### 5.3.1 Problem Formulation

Given a set of critical pairs of nets  $CP = \{(n_i, n_j)\}$ , (critical pairs are pairs of nets between whom coupling constraints are present) and a set of associated bounds  $\{C_{bound}(i, j)\}$ , the problem is to perform channel routing, such that the capacitance  $C(i, j)$  between nets  $n_i$  and  $n_j$ , satisfies

$$C(i, j) \leq C_{bound}(i, j) \quad \forall (n_i, n_j) \in CP \quad (5.1)$$

The bounds on the critical capacitances can be automatically obtained from the performance constraints using the constraint generator PARCAR as described earlier, or can be specified by the user.

Capacitive coupling is taken into account when two segments *cross*, or when two parallel segments run *adjacent* to each other. The specified constraints on coupling capac-

instances will then manifest themselves in the form of *crossover constraints* and *adjacency constraints*. The parasitic constraints are imposed by *mapping the crossover and adjacency constraints into constraints in the VC graph*. This is achieved by *directing some undirected edges, adding some directed edges or increasing the weights of some edges in the VC graph* as depicted in Fig. 5.2.

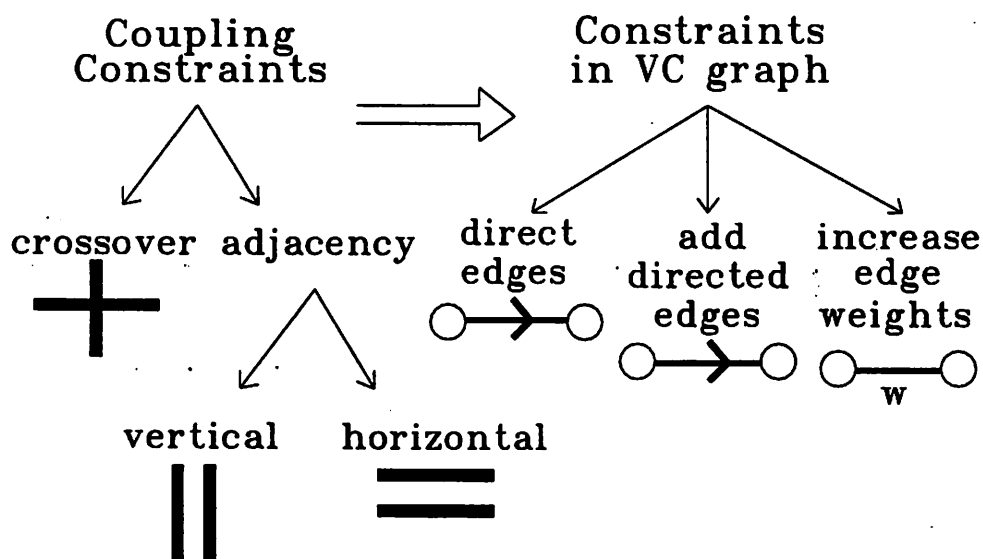


Figure 5.2: The constraint-mapping problem

Sometimes, the pin positions may be such that it is not feasible to meet a specified parasitic constraint, and as a result, one has an overconstrained condition. On the other hand, in some cases, there can be several different choices available while deciding on which constraint to impose on the VC-graph. In the later case, the constraint which causes minimum increase in the length of longest directed path (from source to the sink node) in the VC-graph is chosen. This is referred to as mapping in an *optimal way* and is a heuristic approach for causing minimum increase in channel height because of the constraints (the exact minimization problem is NP complete even without any coupling constraint).

### 5.3.2 The Mapping Algorithms to Eliminate Critical Couplings

Now, the mapping algorithms are presented for the special case, when the bounds

on a set of critical capacitances are “0”. Several important issues can be identified easily if this special case is considered first. The algorithms can be easily generalized to the case when nonzero bounds are specified as discussed in Section 5.3.3.

For this special case, when the bound is zero, the *crossover constraint* is said to be satisfied, if no two segments of the two nets cross. The *adjacency constraint* is said to be satisfied, when each pair of parallel lines of the two nets (a) run at a distance larger than a certain specified influence distance  $ID$  (*separation condition*) OR (b) is shielded by a shield line running between the two lines (*shield condition*).

Any model for capacitance between two parallel lines can only be accurate up to a certain maximum separation between two lines. Hence, coupling is ignored beyond a certain influence distance  $ID$ , whose value will depend on the model being used. A model generator CAPMOD has been developed, which obtains capacitance models for various common configurations in a given process, by performing numerical simulations and fitting analytical expressions based on a partial knowledge of flux components associated with the configurations. Refer to Chapter 6 for a detailed description of the model generator.

The lines which can be used for the purpose of shielding should be specified, and are typically the power/ground lines. Hence, the coupling constraint on the capacitance between a critical pair of nets is satisfied when both the crossover and the adjacency constraints are satisfied.

In the discussion which follows, all distances are measured with respect to center lines of interconnects. Hence, for clarity, nonzero variable widths of lines have not been shown in the figures. Also the top and bottom edges of lines have been shown to be straight lines, although the actual algorithms can handle irregular edges. Since each net is associated with a distinct node in the VC graph, the term *net* itself will be used for the corresponding node in the VC graph. Moreover, some parts of Appendix 4.11 of previous chapter (dealing with minimum and maximum estimates of capacitances) may appear to be repeated here. But, the minimum and maximum estimates addressed in this chapter are made during routing, as opposed to a-priori estimates presented in Appendix 4.11.

### **Crossover Constraints**

First, conditions are derived under which the crossover between two nets (say  $n_1$

and  $n_2$ ) can be avoided. When the two nets do not have common horizontal span, the crossover capacitance will be zero irrespective of where the horizontal segments of the two nets are placed. When they have common span, there are two possible cases. (a) the horizontal segment of  $n_1$  is above that of  $n_2$  and (b) the horizontal segment of  $n_2$  is above that of  $n_1$ . The expressions for crossover capacitance are now derived for these two cases. As illustrated in Fig. 5.3, in case (a) each top pin ( $p_2$ ) of  $n_2$  and each bottom pin ( $p_1$ ) of  $n_1$  lying within the common horizontal span of the two nets, will contribute a crossover, and vice versa for case (b) (note that one vertical segment is used at each pin position).

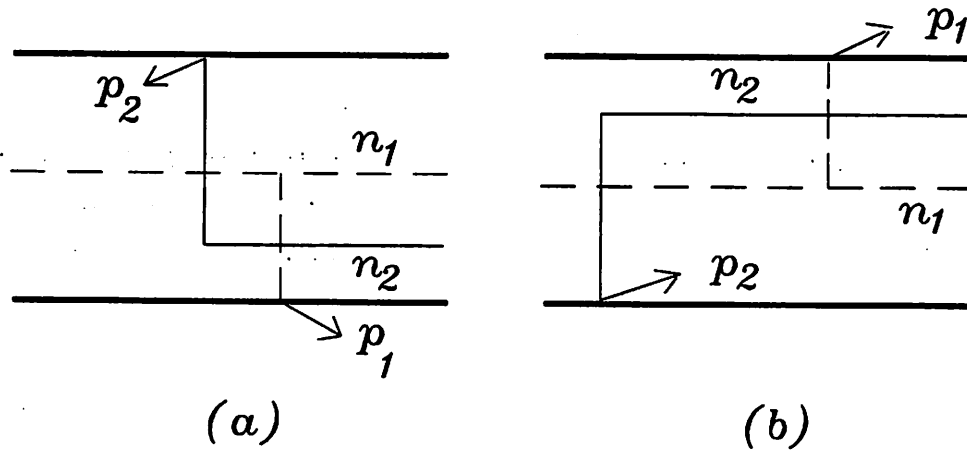


Figure 5.3: Dependence of crossover capacitance on pin positions

Let the number of *top(bottom)* pins of nets  $n_1(n_2)$  in the *common horizontal span* of the two nets be respectively denoted by  $N_t(n_1)$ ,  $N_b(n_1)$ ,  $N_t(n_2)$  and  $N_b(n_2)$ . Let  $C_{c12}$  ( $C_{c21}$ ) be the capacitance contribution of *each* crossover due to the horizontal segment of  $n_1(n_2)$  and a vertical segment of  $n_2(n_1)$ . If  $C_{cross12}$  and  $C_{cross21}$  are respectively the total crossover capacitances in cases (a) and (b) then

CASE(a):

$$C_{cross12} = N_t(n_2) * C_{c12} + N_b(n_1) * C_{c21} \quad (5.2)$$

CASE(b):

$$C_{cross21} = N_t(n_1) * C_{c21} + N_b(n_2) * C_{c12} \quad (5.3)$$

From (5.2) it is obvious that crossover can be avoided by forcing the horizontal segment of  $n_1$  to lie above that of  $n_2$ , if and only if (i)  $N_t(n_2) = 0$ , and (ii)  $N_b(n_1) = 0$

and (iii) it is feasible to have a directed edge from  $n_1$  to  $n_2$  in the VC graph without forming a cycle. Condition (iii) will be satisfied when there is no directed path existing from  $n_2$  to  $n_1$  in the VC graph. The conditions for avoiding the crossover by forcing the horizontal segment of  $n_2$  above  $n_1$  are obtained by interchanging  $n_1$  and  $n_2$  in the above three conditions. Hence, the necessary and sufficient condition for avoiding the crossover between two nets with common horizontal span follows.

**Lemma 5.1** *Let each net be implemented by one horizontal segment and a vertical segment for each of its pins. Then the crossover between two nets  $n_1$  and  $n_2$  can be avoided iff (A)  $N_t(n_2) = N_b(n_1) = 0$ , and there is no directed path from  $n_2$  to  $n_1$  in the VC graph; OR (B)  $N_t(n_1) = N_b(n_2) = 0$ , and there is no directed path from  $n_1$  to  $n_2$  in the VC graph.*

If the above condition for avoiding the crossover is not satisfied, then each unavoidable crossover between the two specific nets has to be shielded. This is possible when three layers of interconnects are available[16]. For example, when the horizontal and vertical segments in the channel are routed in MET1 and MET2, then the interconnect in MET1 can be shifted for a short length to POLY in the vicinity of the crossover, and a grounded MET1 plate used to shield the crossover. A maze router has to be used in the post processing stage to make the necessary ground connection to the shield plates. The algorithm for mapping the crossover constraints into VC graph now follows. Conditions (A) and (B) in Lemma 5.1, are referred to as **condA** and **condB** respectively.  $LDP(Top, Bottom)$  denotes the length of longest directed path from the source node to the sink node in the VC graph.

**Algorithm 1 (Mapping of Crossover Constraints)**

```

if ((condA is false) AND (condB is false)) then
    crossover needs to be shielded;
else if ((condA is true) AND (condB is false)) then
    direct  $n_1$  to  $n_2$  in the VC graph (if already not directed);
else if ((condB is true) AND (condA is false)) then
    direct  $n_2$  to  $n_1$  in the VC graph (if already not directed);
else
    direct to minimize  $LDP(Top, Bottom)$ ;

```

In the above discussion, it was assumed that there is one horizontal segment for each net and one vertical segment for each pin. However, it may be useful to have a necessary condition on the pin positions to avoid crossover between two nets when such restrictions are removed. Appendix 5.10.

**Lemma 5.2** *A necessary condition for avoiding the crossover between two nets  $n_1$  and  $n_2$  (irrespective of how they are routed inside the channel) is (A)  $N_t(n_2) = N_b(n_1) = 0$ , OR (B)  $N_t(n_1) = N_b(n_2) = 0$ .*

**Proof:** Appendix 5.10.

As expected, the necessary condition is less strong than that of the restrictive case assumed in Lemma 5.1. If the necessary condition of Lemma 5.2 is not satisfied, then there is no way the crossover between the nets can be avoided, and hence, it has to be shielded during post processing. However, if that condition is satisfied, i.e. if  $N_t(n_2) = N_b(n_1) = 0$ , OR  $N_t(n_1) = N_b(n_2) = 0$ , but due to one or more vertical constraints, the conditions of Lemma 5.1 are not satisfied, then one may direct the necessary edge to avoid crossover, even if that forms a cycle and break the cycle afterwards using a maze router as is done in [82].

It is to be noted that, the necessary condition for avoiding the crossover between two nets is the same as that for routing the two nets in the same layer. There has been some work in developing algorithms for checking the positions of fixed pins in a layout to make sure that the routing can be achieved on a single layer. Such algorithms have been described in [105] which also includes other related references. However, it is not possible to directly apply such algorithms to the case one is interested here, since the channel has floating pins on the left and right edges. Moreover, the result of Lemma 2 gives a compact condition which can be used conveniently during pin assignment and global routing of analog circuits to satisfy (if possible) these conditions.

### **Adjacency Constraints between Vertical Segments**

Each fixed pin on the channel edge is associated with a vertical segment. Hence, two vertical segments can run parallel to each other within the influence distance  $ID$ , when the corresponding pins are within the same distance. In that case, if the segments have

common span, then since the separation between the segments is fixed by the pin positions, one needs to satisfy only the *shield condition*. There are two possible cases regarding the relative positions of the two pins within distance  $ID$ : (a) pins on the same edge of the channel (b) pins on opposite edges of the channel. In each case, conditions to satisfy the adjacency constraint will now be identified.

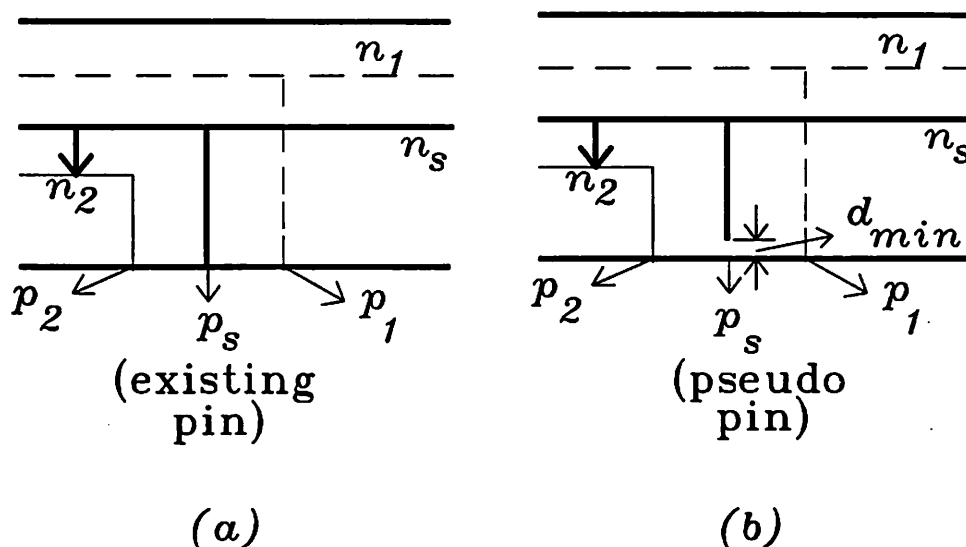


Figure 5.4: Shielding achieved using one vertical segment for CASE 1

CASE 1: When the two pins are on the same edge of the channel, the associated vertical segments are going to run parallel to each other for some finite distance. Hence, to shield the coupling between the two lines, a vertical segment of a shield net has to be inserted between the two segments (if there is space), and if possible the height of it should be made larger than that of the segment associated with either pin as shown in Fig. 5.4 (a). If there is already a pin  $p_s$  of a shield net  $n_s$  placed between the two pins, then this constraint can be imposed by directing an edge from  $n_s$  to  $n_1$  or  $n_s$  to  $n_2$  if feasible without forming a cycle. Even if there is no such pin  $p_s$ , an imaginary pin  $p_s$  can be created. In that case, if the vertical segment of the shield segment is not allowed to touch the channel edge because of the presence of some other object, a minimum gap can be created as shown in Fig. 5.4 (b). The term *intermediate* will be used to refer to a pin (on either edge) lying between the  $x$  coordinates of  $p_1$  and  $p_2$ .

Condition **cond1A** is said to be satisfied *if there is a directed edge between a shield net (having a real/imaginary intermediate pin) and either  $n_1$  or  $n_2$* . If it is not possible to



shield the coupling using one vertical segment, one can try two vertical segments (of two shield nets as shown in Fig. 5.5 (a), or of one shield net as shown in Fig. 5.5 (b)), whose combined span shields the common span of the nets  $n_1$  and  $n_2$ . Condition **cond1B** is said to be satisfied if *there is a directed edge from a shield net having a real/imaginary intermediate pin on the bottom edge, to another shield net having a real/imaginary intermediate pin on the top edge*. Condition **cond1C** is said to be satisfied if *a shield net has real/imaginary intermediate pins both on the bottom and top edges*. Note that satisfaction of **cond1C** does not depend on the direction of any edge.

There is no need to look for more than two vertical segments for shielding, since otherwise two of them have to have the associated intermediate pins on the same edge and as a result, the vertical span of one contained in another. This will mean that one of the segments is redundant.

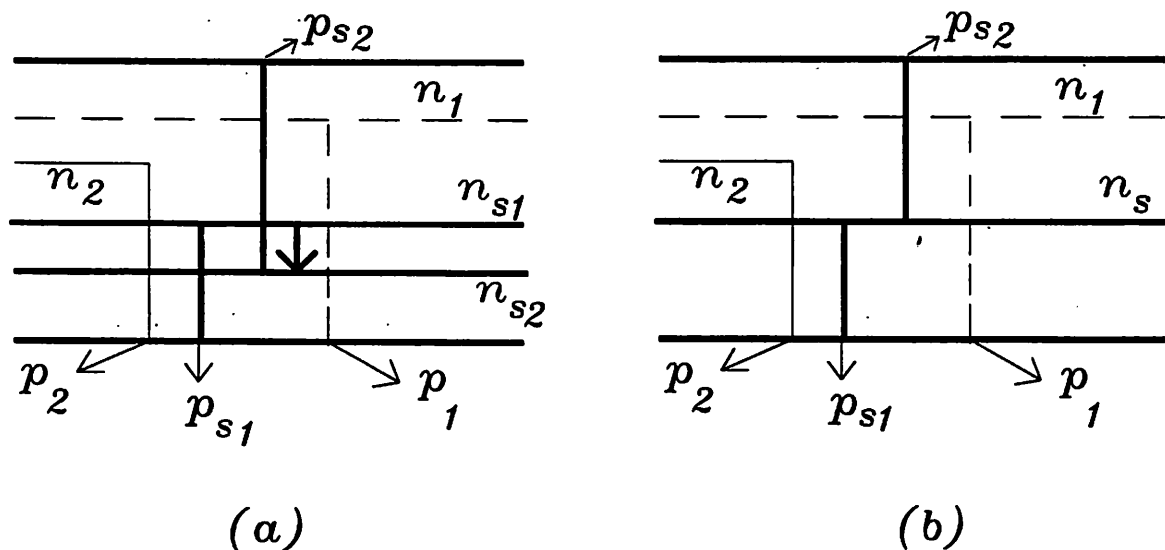


Figure 5.5: Shielding achieved using two vertical segments for CASE 1

**CASE 2:** When the two pins  $p_1$  and  $p_2$  are on different edges of the channel, then it may be possible to prevent them from having a common vertical span, by forcing the horizontal segment of net with the top pin to lie above that of the net with the bottom pin (as shown in Fig. 5.6(a)). If the two nets do not have common horizontal span (hence no edge between the nets in the VC graph), then this requires *adding* a directed edge to the VC graph. Condition **cond2A** is said to be satisfied if *there is a directed edge from the net containing the top pin to the net containing the bottom pin in the VC graph*. When the two

vertical segments have common span, then it may be possible to shield the coupling using one or two vertical segments of one or two shield nets just as in the case when pins are on the same edge. In a manner similar to **cond1A** , **cond1B** and **cond1C** respectively, conditions **cond2B** , **cond2C** and **cond2D** can be defined (illustrated in Fig. 5.6(b), 5.6(c) and 5.6(d)).

For each of the above two cases, the algorithm first checks if at least one of the conditions of shielding is satisfied due to the constraints already present in the VC graph. If not, then an edge is directed to satisfy one of the conditions (if possible without forming a cycle). If a choice exists, the operation causing minimum increase in  $LDP(top, bottom)$  is performed.

**Algorithm 2 (Mapping of Adjacency Constraints Between Vertical Segments)**

```

for each pin pair  $(p_1, p_2)$ , such that  $p_1 \in n_1, p_2 \in n_2$  and  $|x(p_1) - x(p_2)| \leq ID$ , begin
  if  $p_1$  and  $p_2$  are on the same channel edge begin
    if (cond1A is TRUE) OR (cond1B is TRUE) OR (cond1c is TRUE)
      the pair  $(p_1, p_2)$  is already shielded;
    else direct an edge without forming a cycle in the VC graph to satisfy cond1A or
cond1B if possible;
      (if not possible constraint can not be met)
      (if more than on choice minimize  $LDP(Top, Bottom)$ )
    end
    else if (cond2A is TRUE) OR (cond2B is TRUE) OR (cond2c is TRUE) OR
(cond2D is true)
      the pair  $(p_1, p_2)$  is already shielded;
    else direct an edge (or add a directed edge) without forming a cycle in the VC graph
      to satisfy cond2A or cond2B or cond2C if possible;
      (if not possible constraint can not be met)
      (if more than on choice minimize  $LDP(Top, Bottom)$ )
    end
  end
end

```

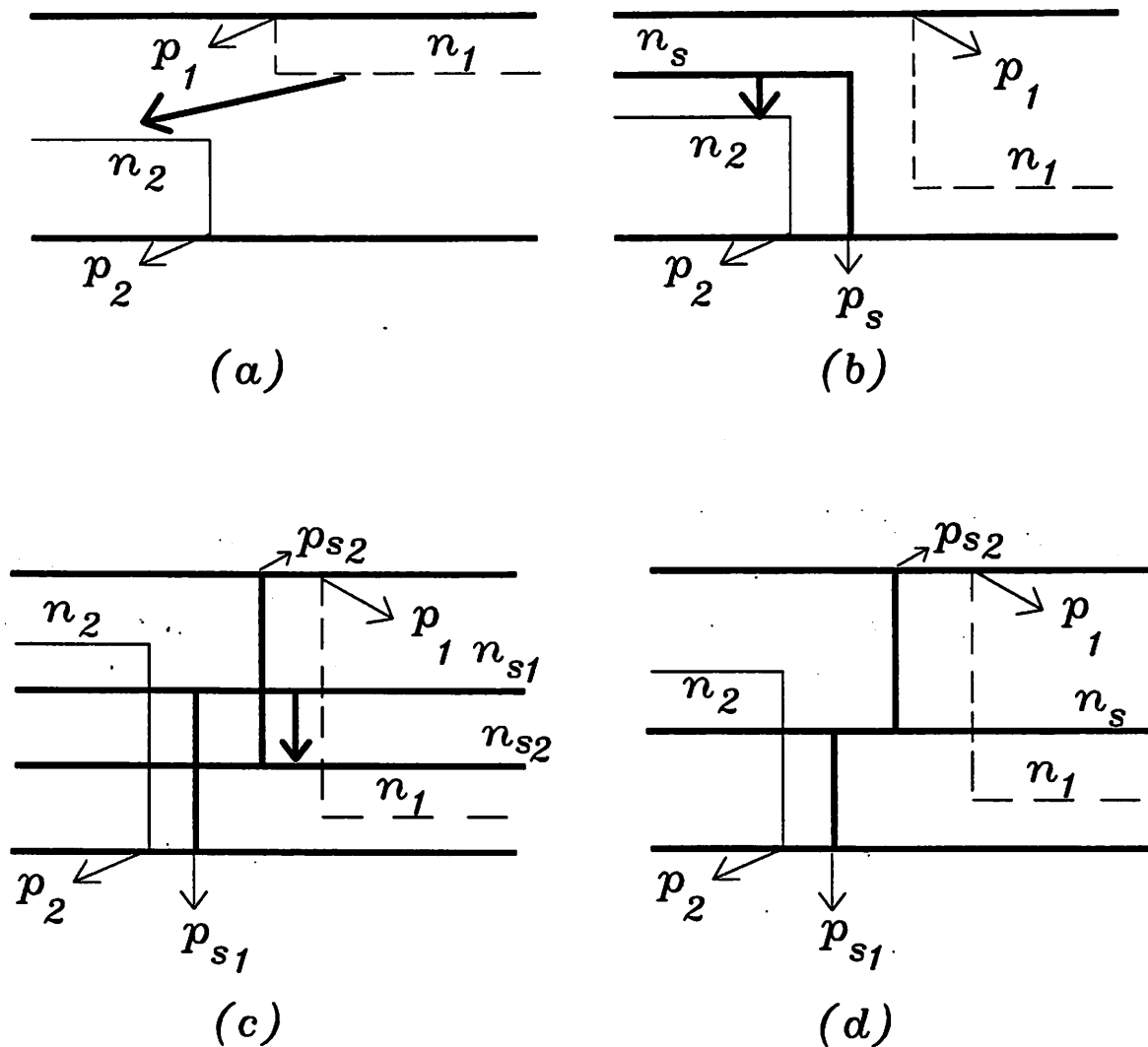


Figure 5.6: Illustration of conditions in CASE 2

### Adjacency Constraints Between Horizontal Segments

If two nets  $n_1$  and  $n_2$  have common horizontal span, then their respective horizontal segments will run parallel to each other. Hence, if they have critical coupling constraint, as mentioned earlier, the two horizontal segments (say  $seg_1$  and  $seg_2$ ) have to satisfy the *shield condition* OR the *separation condition*. The separation condition can be easily satisfied by setting the weight of the edge between the nodes (representing the two nets) in the VC graph to a large value (say  $ID$ ). The weight of the edge represents the minimum separation

between the horizontal segments. However, this can give rise to waste in area if segments of other nets do not run between  $seg_1$  and  $seg_2$ . Hence, first try to satisfy the shield condition by constraining the positions of one or more available shield nets between  $seg_1$  and  $seg_2$ , and only if that is not possible, the edge weight is set to  $ID$ .

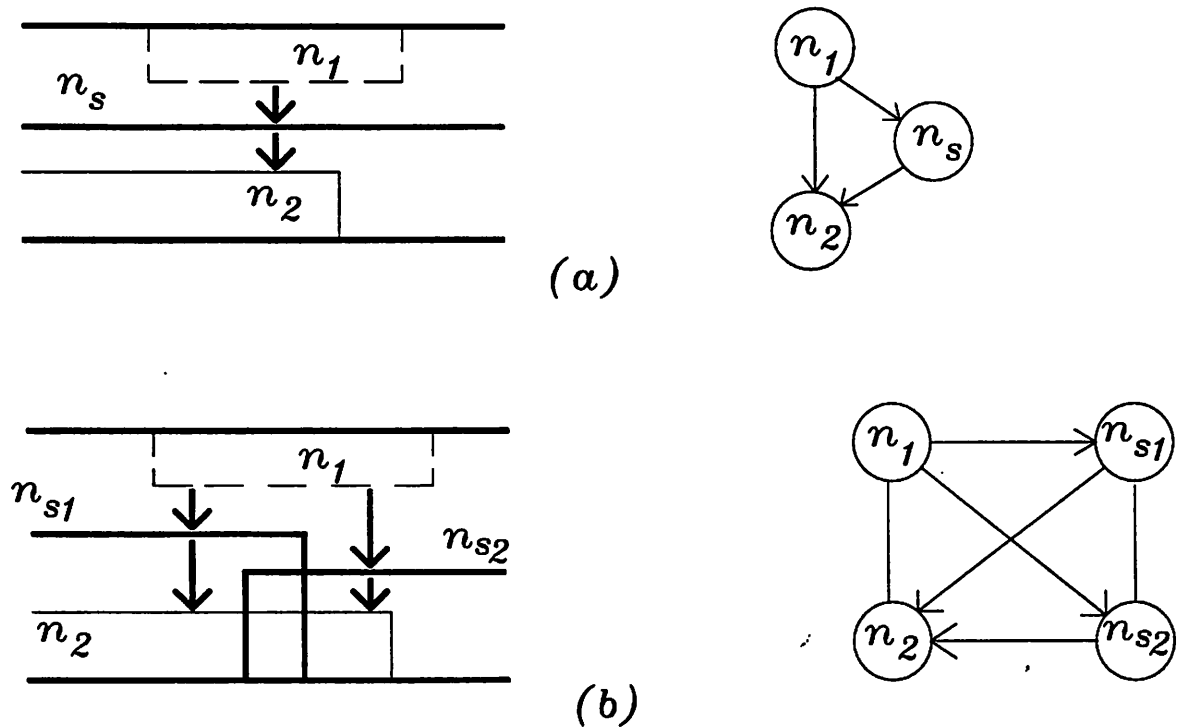


Figure 5.7: Constraining the horizontal segments of shield nets

The horizontal segment  $seg_s$  of a shield net  $n_s$  can be used for shielding (fully or partially) the coupling between the horizontal segments  $seg_1$  and  $seg_2$  if the horizontal span of  $seg_s$  intersects the common horizontal span of  $seg_1$  and  $seg_2$ . Let the set of all shield nets satisfying the above condition be denoted by  $N_s(n_1, n_2)$ . Then the goal will be to constrain the positions of one or more of the nets belonging to  $N_s(n_1, n_2)$  so that the union of the horizontal spans of these constrained nets includes the common horizontal span of  $seg_1$  and  $seg_2$ . The horizontal segment of a net  $n_s$  is constrained to lie between those of  $n_1$  and  $n_2$  if there is a pair of *directed* edges one of them originating from  $n_1$  (or  $n_2$ ) and terminating on  $n_s$ , and the other originating from  $n_s$  and terminating on  $n_2$  (or  $n_1$ ) as shown in Fig. 5.7(a). At any stage in the constraint mapping process, the set of all such constrained shield nets is denoted by  $N_{sc}(n_1, n_2)$ .

If  $span(n)$  denotes the interval representing the horizontal span of a net  $n$ , then the *unshielded common span* of the pair  $(n_1, n_2)$  denoted by  $ucs(n_1, n_2)$  is defined as

$$ucs(n_1, n_2) = [span(n_1) \cap span(n_2)] - \bigcup_{n_s \in N_{sc}(n_1, n_2)} span(n_s) \quad (5.4)$$

The unshielded common span may be a collection of disconnected intervals. The length of the  $ucs(n_1, n_2)$ , denoted by  $lucs(n_1, n_2)$  is the sum of the lengths of each of the individual intervals in  $ucs(n_1, n_2)$ .

During the process of mapping the coupling constraints into VC graph, when there are more than one shield net to choose from, the shield net whose horizontal span has the maximum intersection with the  $ucs(n_1, n_2)$  is chosen. If after constraining the position of that shield net,  $lucs(n_1, n_2)$  is not zero, the process is repeated. If no more shield net is available at any stage and  $lucs(n_1, n_2)$  is still not zero, then the weight of the edge  $(n_1, n_2)$  in the VC-graph is set to  $ID$ . At any stage in the mapping process, a good lower bound for the separation between the horizontal segments of  $n_1$  and  $n_2$  should also be computed taking into account the constraints already present in the VC graph. If the estimated minimum separation exceeds  $ID$ , then even if the shield condition is not satisfied, the coupling can be assumed to be eliminated. The length of the longest directed path in the VC graph from  $n_1$  to  $n_2$  ( $LDP(n_1, n_2)$ ) will always give a lower bound on the separation between the two associated horizontal segments. However, it can be a very loose lower bound. For example, consider the case in Fig. 5.7(b). When net  $n_{s1}$  is constrained to lie between  $n_1$  and  $n_2$ , then the length of the longest directed path between  $n_1$  and  $n_2$  becomes twice the *minimum net-to-net separation* ( $d_{min}$ ). When net  $n_{s2}$  is constrained, it remains the same. However, since  $n_{s1}$  and  $n_{s2}$  have common horizontal span, irrespective of how  $n_{s1}$  and  $n_{s2}$  are placed with respect to each other, the minimum separation between  $n_1$  and  $n_2$  is  $3d_{min}$ . Hence, it is quite evident that a significantly better lower bound on the minimum separation can be obtained by considering all possible relative placements of the nets which are *constrained* to lie between the nets  $n_1$  and  $n_2$ .

Let the set of nets (not necessarily shield nets) which are constrained to lie between  $n_1$  and  $n_2$  be denoted by  $N_c(n_1, n_2)$ . Let  $S_{vc}(n_1, n_2)$  be the set of VC graphs obtained by considering all feasible directions (which do not lead to cycles) of the undirected edges existing between any two nodes representing nets in  $N_c(n_1, n_2)$ . Then a lower bound on the minimum separation  $d_{low}(n_1, n_2)$  between horizontal segments of nets  $n_1$  and  $n_2$  is computed as

$$d_{low}(n_1, n_2) = MIN \{LDP(n_1, n_2) \text{ in VC graph}; VC \text{ graph} \in S_{vc}(n_1, n_2)\} \quad (5.5)$$

**Algorithm 3 (Mapping of Adjacency Constraints Between Horizontal Segments)**

```

while ( $lucs(n_1, n_2) > 0$  AND ( $d_{low}(n_1, n_2) < ID$ )) begin
  choose  $n_s \in N_s(n_1, n_2)$  having maximum common span with  $ucs(n_1, n_2)$ ;
  if ( $length(n_s \cap ucs(n_1, n_2)) > 0$ ) begin
    if feasible without forming cycle, constrain  $n_s$  by directing two edges,
    one from  $n_1$  (or  $n_2$ ) to  $n_s$ , and the other from  $n_2$  (or  $n_1$ ) to  $n_s$ ;
    (if more than one choice minimize  $LDP(Top, Bottom)$ )
    if  $n_s$  is constrained
      add  $n_s$  to  $N_{sc}(n_1, n_2)$  and also  $N_c(n_1, n_2)$ ;
  end
end
else
  set weight of edge( $n_1, n_2$ ) to  $ID$ ; quit.
end

```

After the crossover and adjacency constraints are imposed on the vertical-constraint graph, there may still be some undirected edges in the graph which have to be directed. This can be achieved by using a gridless channel routing algorithm as described in [95] which directs the undirected arcs to find a nearly optimal placement of the horizontal segments to minimize channel height. Since the algorithm is similar to that presented in [95], it is not described here. The minimization of channel height is thus subjected to the set of parasitic constraints embedded in the VC graph.

### 5.3.3 Meeting Nonzero Bounds on Coupling Capacitances

We shall now describe how the algorithms presented in Section 5.3.2 can be extended to handle *nonzero* bounds on a set of critical coupling capacitances. Let  $C_{bound}$  denote the maximum allowed capacitance between two specific nets  $n_1$  and  $n_2$ . While mapping the bounding constraint on the capacitances to constraints in the VC graph, it becomes necessary to dynamically estimate a maximum and a minimum value for the capacitance between the two nets taking into account the constraints already present in the VC graph

at any stage of routing. This is because if the maximum estimated capacitance is below the bound, no more effort is required for meeting the constraint associated with that particular pair of nets. If however, the bound is less than the minimum estimate, then shielding has to be done during post processing (may be necessary for unavoidable crossovers as explained earlier).

The crossover capacitance is determined by the relative positions of the horizontal segments of the two nets. From (5.2) and (5.3), the maximum and minimum estimates for the crossover capacitance denoted by  $C_{cross\_max}$  and  $C_{cross\_min}$  are given by

$$C_{cross\_max} = MAX(C_{cross\_12}, C_{cross\_21}) \quad (5.6)$$

$$C_{cross\_min} = MIN(C_{cross\_12}, C_{cross\_21}) \quad (5.7)$$

Let the per-unit-length capacitance between two adjacent parallel segments of nets  $n_1$  and  $n_2$  separated by a distance  $d$  in the layer corresponding to the vertical and horizontal segments be denoted by  $C_v(d)$  and  $C_h(d)$  respectively. As mentioned earlier, the models are obtained for a given process from a model generator. Let the set of pin pairs  $(p_1, p_2)$ , such that  $p_1 \in n_1$ ,  $p_2 \in n_2$ , and  $|x(p_1) - x(p_2)| < ID$ , and whose corresponding vertical segments are not already shielded (refer to the conditions presented in Algorithm 2) be denoted by  $P_{us}$ . The vertical segments associated with each such pair can run at the most a distance equal to the channel height. Although the channel height is not exactly known during routing, the channel density can be taken as a good estimate, because good channel routers can achieve a channel height very close to the channel density. Hence, the maximum coupling between the vertical segments of the nets is estimated as

$$C_{v\_max} = \sum_{(p_1, p_2) \in P_{us}} (\text{channel density}) * C_v(|x(p_1) - x(p_2)|) \quad (5.8)$$

The minimum estimate is taken as zero. The maximum estimate of the capacitance between two horizontal segments of the nets is obtained by considering the length of the unshielded common span (obtained from (5.4)) and a separation equal to its lower-bound estimate (obtained from (5.5)).

$$C_{h\_max} = lucs(n_1, n_2) * C_h(d_{low}(n_1, n_2)) \quad (5.9)$$

The minimum estimate is again taken as zero. Hence the maximum and minimum estimates of the total capacitance between the nets are  $C_{max} = C_{cross\_max} + C_{h\_max} + C_{v\_max}$ ;

$C_{min} = C_{cross\_min}$ . If  $C_{bound} < C_{min}$ , then it means that the minimum crossover capacitance itself exceeds the bound, and the unavoidable crossovers should be shielded in the post processing stage (we assume technology permits it).  $C_{min}$  can then be set to zero. Hence, the following cases are considered.

(CASE 1)  $C_{bound} > C_{max}$  : In this case the constraint is already met. No additional effort is required for the pair  $(n_1, n_2)$ .

(CASE 2)  $C_{cross\_min} < C_{bound} < C_{cross\_max}$  : This implies that one of the two possible orderings of the horizontal segments of the two nets will violate the constraint. If the other ordering does not form a cycle, then the edge  $(n_1, n_2)$  in VC graph is directed accordingly and  $C_{cross}$  is set to  $C_{cross\_min}$ . If not, the other direction is set and the resulting crossovers have to be shielded during post processing ( $C_{cross}$  is set to zero).

(CASE 3)  $C_{cross\_max} < C_{bound} < C_{max}$  : This implies there is a possibility of meeting the constraint for both possible orderings of the horizontal segments of the two nets. If both directions of the edge  $(n_1, n_2)$  in the VC graph are feasible without forming a cycle, then the direction causing minimum increase in the  $LDP(Top, Bottom)$  should be chosen.  $C_{cross}$  is set to either  $C_{cross\_min}$  or  $C_{cross\_max}$  accordingly. The procedure followed for directing the edges in the above two cases will be referred to as **process\_crossovers()** .

Once the crossover capacitance is known, the pin pairs of the two nets can then be processed as stated in Algorithm 2 (the procedure referred to as **process\_pin\_pair()** in this section), and after processing each pin pair, the maximum estimate for capacitance can be updated (using the information about the crossover capacitance and the pin pairs already shielded). The maximum estimate can be compared with the bound to check if the constraint is already met. After processing all the pin pairs, if the constraint is still not met, a bound  $C_{h\_bound}$  on the capacitance between horizontal segments is calculated.

In Algorithm 3, the goal was to shield completely the common horizontal span by constraining one or more shield nets, and if that was not possible, to set the minimum separation between the segments to  $ID$ . Now the goal will be to satisfy  $C_{h\_max} < C_{h\_bound}$  by constraining one or more shield nets and if that is not possible, the weight of the edge  $(n_1, n_2)$  should be set to  $d_{sep}$ , where  $d_{sep}$  is the minimum separation meeting the bound, and can be obtained from the following relation.

$$C_h(d_{sep}) * lucs(n_1, n_2) = C_{h\_bound} \quad (5.10)$$



This modified version of Algorithm 3 is referred to as **process\_horizontal\_segments()**. After all the parasitic constraints are imposed on the VC graph, there can still be some undirected edges. As mentioned earlier, they are directed in a manner similar to that described in [95], and this procedure is referred to as **direct\_remaining\_edges()**. The overall algorithm for meeting the bounds on coupling capacitances now follows.

**Algorithm 4 (Routing Driven by Nonzero Bounds on Coupling Capacitances)**

```

for each net pair  $(n_1, n_2) \in CP$  begin
  while  $(C_{max}(n_1, n_2) > C_{bound}(n_1, n_2))$  begin
    process_crossovers() ;
    Update  $C_{max}(n_1, n_2)$ ;
    for each pin pair  $(p_1, p_2)$ , such that  $p_1 \in n_1, p_2 \in n_2$ , and  $|x(p_1) - x(p_2)| \leq ID$  begin
      process_pin_pair()
      Update  $C_{max}(n_1, n_2)$ ;
    end ;
    Compute  $C_{h\_bound}$  ;
    process_horizontal_segments();
  end
end
direct_remaining_edges();

```

## 5.4 Imposing the Matching Constraints on Channel Routing

The need for matching constraints is discussed in Section 4.5. This section deals with how to impose the matching constraints on channel routing.

### 5.4.1 Defining Symmetrical Channels

As was mentioned earlier, a layout can be routed completely using channel routing, if the placement of components forms a slicing structure. Suppose a slicing structure is

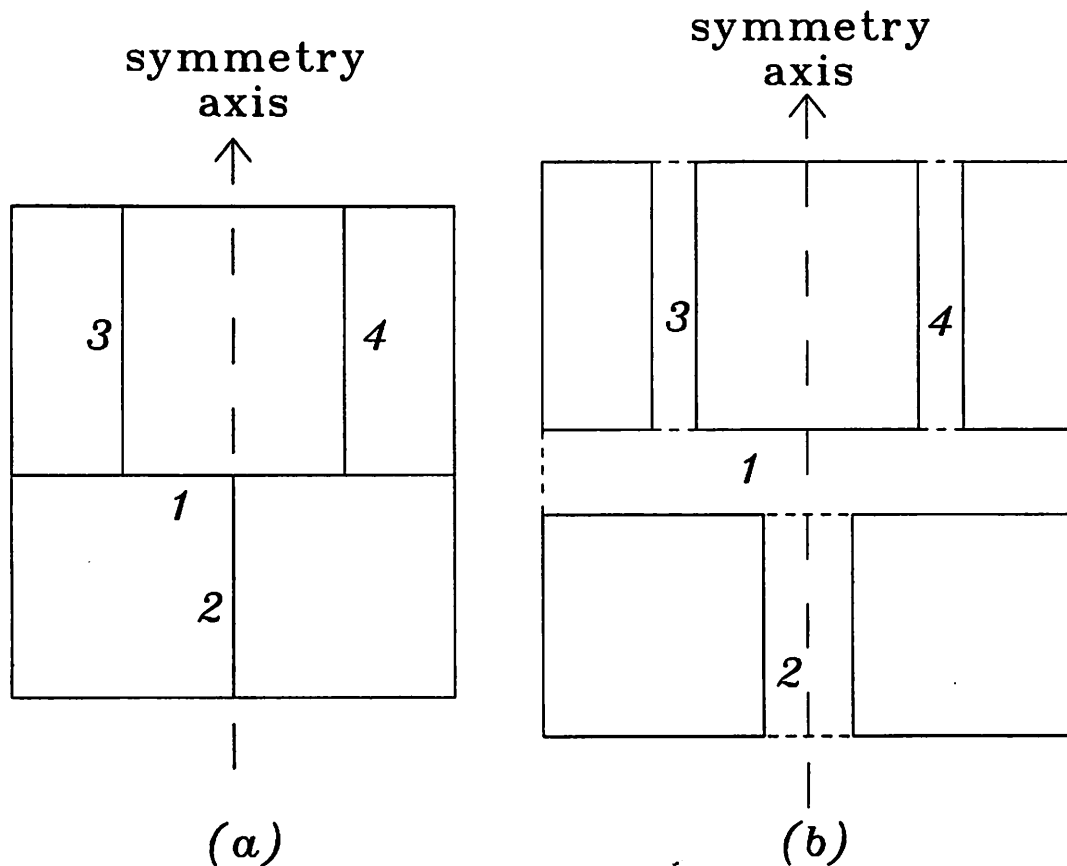


Figure 5.8: A symmetrical slicing structure and the associated channels

considered, which has a mirror symmetry around an axis as shown in Fig. 5.8. When the corresponding channels are defined, a channel can belong to one of the following three categories: (a) the axis of symmetry does not pass through the channel (as for channels 3 and 4 in Fig. 5.8). Such channels occur in symmetrical pairs. If pin assignment and global routing are also symmetrical, only one channel needs to be routed, and the other can be obtained by flipping. (b) the axis of symmetry passes through the channel perpendicular to the top/bottom edges (as for channel 1 in Fig. 5.8). Such channels are said to have *lateral symmetry*.

(c) the axis of symmetry passes through the channel parallel to the top/bottom edges (as for channel 2 in Fig. 5.8). Such channels are said to have *longitudinal symmetry*. Some layouts can have symmetry requirements only in particular portions. In that case the following discussion applies to only the symmetrical portions of the layout.

### 5.4.2 Routing a Channel with Lateral Symmetry

In this paper only channels with lateral symmetry are considered. However, some of the issues addressed here apply to longitudinal symmetry as well. We also assume that the pin assignment and global routing have mirror symmetry with respect to the nets to be matched, which is usually a valid assumption for differential circuits.

#### The Problem

Two nets in any matched pair can have either (a) nonintersecting horizontal spans or (b) intersecting horizontal spans. First, let the case be considered, when the nets in each matched pair have nonintersecting horizontal spans as shown in Fig. 5.9 (a).  $(A, \bar{A})$  and  $(B, \bar{B})$  are the two matched pairs in this example and the respective pin positions have mirror symmetry as shown. Hence, if the positions of horizontal segments of the two nets in each pair are forced to be same, then the resistances and capacitances of the nets as well as the direct and cross coupling capacitances with respect to other nets (as discussed in Section 4.5) are matched. This constraint can be easily imposed on the channel routing algorithm, by considering the nets in each pair as a single net with a discontinuous horizontal span. However, there can be nets like  $C$  in the channel which do not have any matching requirement with respect to any other net. If the pin placement of such a net is symmetric around the axis, it is obvious that the coupling between that net and the nets in each matched pair will automatically be matched. However, if the pin assignment of such a net is not symmetric, then a strong coupling constraint has to be imposed between that net and the matched nets to eliminate any coupling and the associated asymmetry. Hence, it is recommended that pin assignment of such nets be made symmetric if it is not too difficult to do so.

Now let the case be considered, when the nets in at least one of the matched pairs have intersecting horizontal spans as shown in Fig. 5.9 (b) for the pair  $A$  and  $\bar{A}$ . Since each net in that pair has to cross the axis, it is easy to see in this case that neither net in the pair can be obtained by mirror imaging the other net, since otherwise one gets a short circuit at the axis. If one horizontal segment is used for each net, then line resistances and line-to-ground capacitances can be reasonably matched by forcing the horizontal

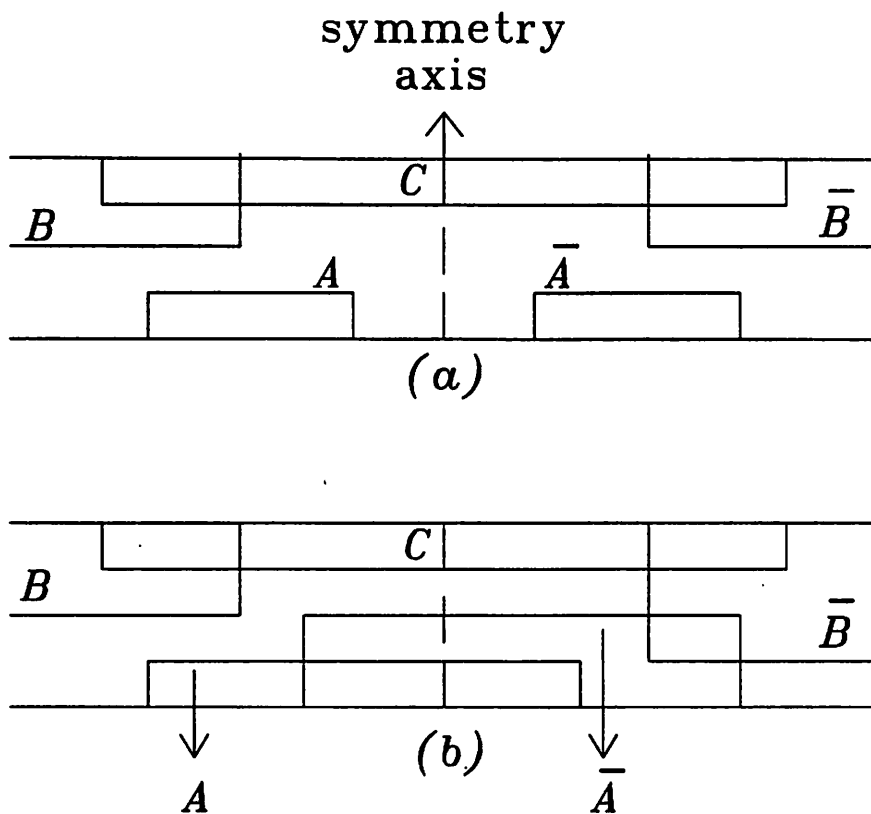


Figure 5.9: (a) Nonoverlapping and (b) Overlapping matched pairs

segments of the two nets to lie next to each other. However, there can be gross mismatch in coupling capacitances between the pair of nets and other nets as shown. Moreover, this may also induce asymmetry into positions of other matched pairs (such as  $(B, \bar{B})$  as shown), if minimum area is to be achieved.

### Achieving Almost-Perfect Mirror Symmetry

As observed, perfect-mirror symmetry cannot be achieved for matched pairs having intersecting horizontal spans since each net has to cross the axis of symmetry. Now, a technique is presented which achieves mirror symmetry with respect to the matched pairs in all regions except a thin strip around the axis (whose thickness will be defined shortly) as shown in Fig. 5.10. Leaving aside that thin strip, on each side of the axis, two horizontal segments are used for the two nets which are forced to remain next to each other, and their positions interchanged on the other side of the axis. The pairs of segments on the two sides

can be connected by a connector which has the shape of "I" in the top view as shown, and hence will be called an "I-connector". As shown in the enlarged view of the connector, the connection of one of the nets is realized in the same layer and that of the other net in another layer using two vias. The horizontal extension of the connector on each side of the axis and the vertical extension are determined by the via-to-line separation denoted by  $d_{vl}$  (measured from center-line to center-line) as shown in Fig. 5.10.

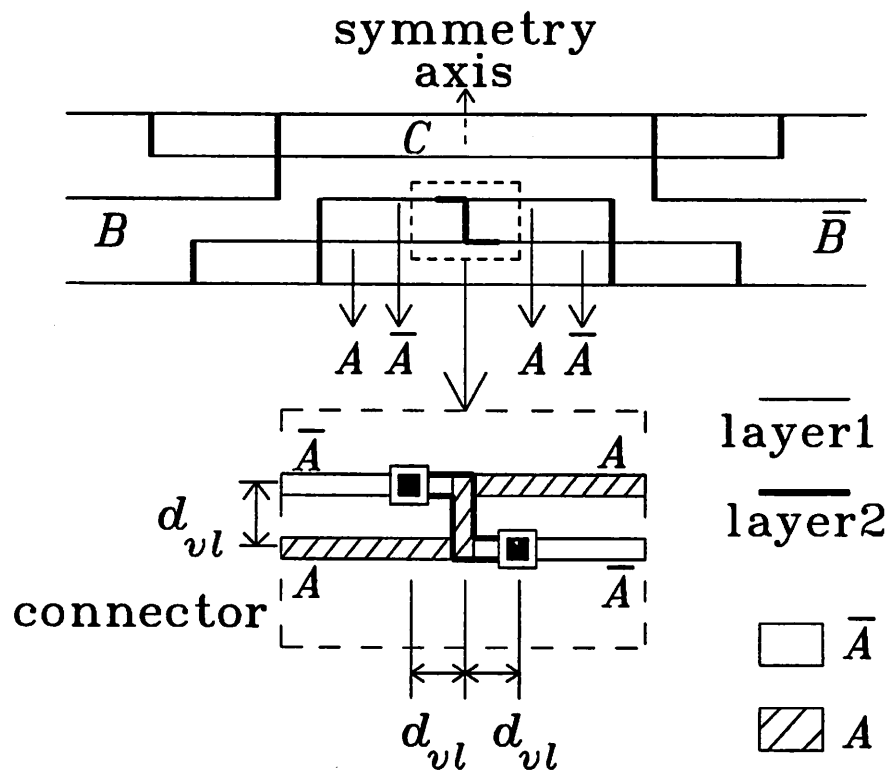


Figure 5.10: Almost-perfect mirror symmetry using I-connectors

In the configuration represented in Fig. 5.10, the resistances and capacitances of the two nets will have a small mismatch because of the vias in one of the nets. If this mismatch is very critical, two more dummy vias can be introduced in the other net as shown in Fig. 5.11. As shown in the enlarged view of this more complicated connector, the resistances and capacitances of the two nets are matched, because for each net, the connector introduces the same length of interconnect and the same number of corners in each layer, besides using the same number of vias. The horizontal extension of the connector on each side of the axis is determined by  $d_{vl}$  (via-to-line separation) and the vertical extension

determined by  $d_{vv}$  (via-to-via separation) as shown in Fig. 5.11.

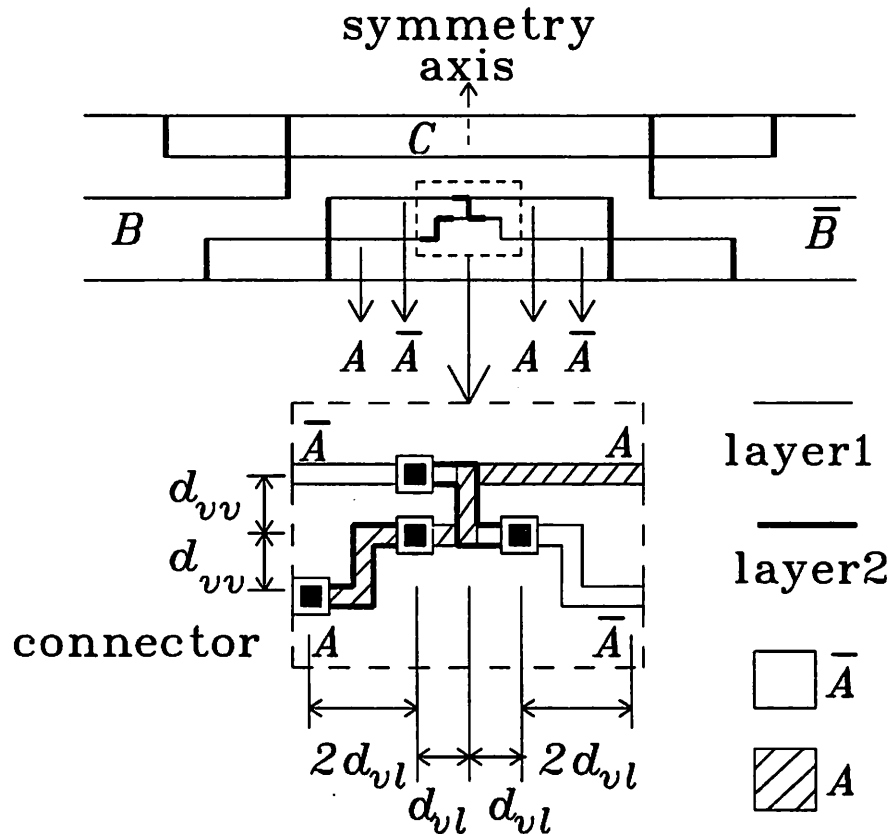


Figure 5.11: Almost-perfect mirror symmetry using I-connectors

As is evident from Fig. 5.10 and Fig. 5.11, since the matched pairs of nets have almost-perfect mirror symmetry, it would result in good matching of direct-coupling and cross-coupling capacitances between the matched pair ( $A, \bar{A}$ ) and the other matched pairs. If a net such as  $C$  (which does not belong to any matched pair) has symmetric pin positions, then the matching will again be good, otherwise the net  $C$  has to be kept away by imposing a tight coupling constraint.

We shall now describe how matching illustrated in Fig. 5.10 can be realized in channel routing (more complicated matching depicted in Fig. 5.11 can be handled in a similar way). The pair of nets in each matched pair is considered as a single net (called a *merged net*) with a nonuniform width as shown in Fig. 5.12 (a), where  $w_A = w_{\bar{A}}$  is the width of each net and  $d_{min}$  is the minimum edge-to-edge separation between the nets. After the routing is completed the two nets can be separated from the merged net. Let the

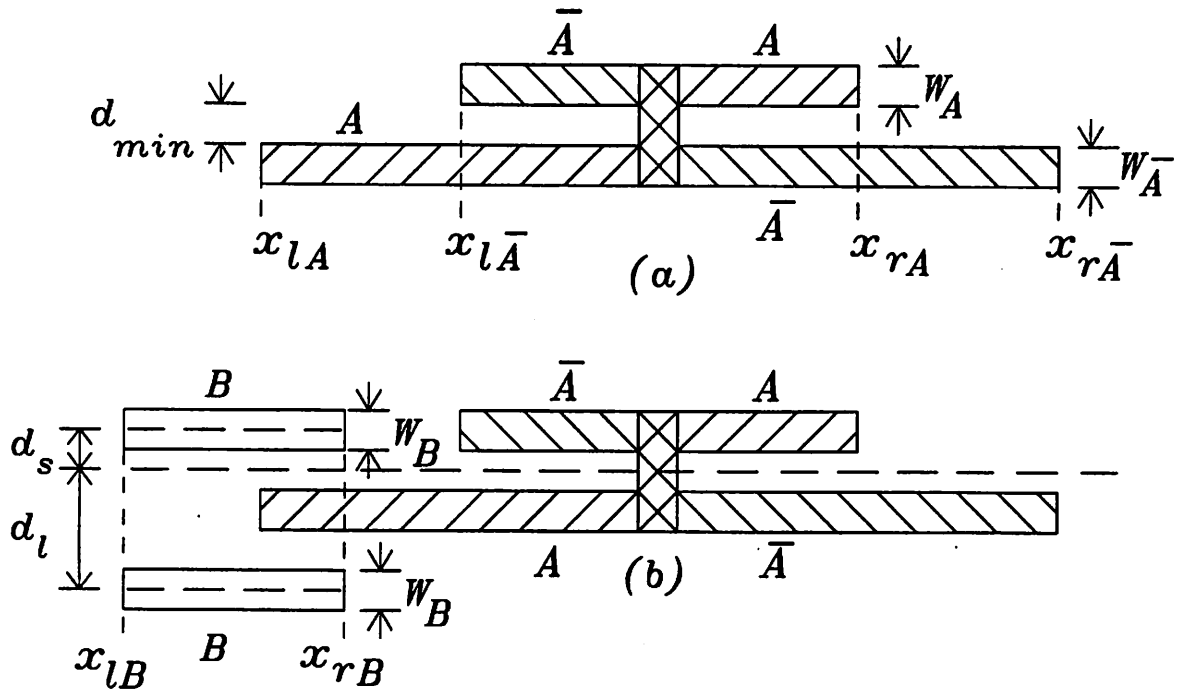


Figure 5.12: Nonuniform width of the special net

matched nets  $A$  and  $\bar{A}$  be merged as shown in Fig. 5.12 (a). Let  $x_{lA}$  and  $x_{rA}$  denote the  $x$  coordinates of leftmost and rightmost pins of net  $A$  respectively, and let  $x_{l\bar{A}}$  and  $x_{r\bar{A}}$  denote equivalent quantities for net  $\bar{A}$ . Symmetry of pin positions implies that if  $x_{lA}$  and  $x_{l\bar{A}}$  are specified,  $x_{rA}$  and  $x_{r\bar{A}}$  are determined. Hence as far as relative positions of these coordinates are concerned, there are two possible cases: (a)  $x_{lA} < x_{l\bar{A}}$ , (b)  $x_{lA} > x_{l\bar{A}}$ . For each of these cases one can have two possible situations regarding the relative positions of nets  $A$  and  $\bar{A}$ , i.e.  $A$  can be either above or below  $\bar{A}$  on the left side of axis. Only the subcase is considered, in which  $A$  is above  $\bar{A}$  on the left side of the axis and  $x_{lA} < x_{l\bar{A}}$ , as shown in Fig. 5.12 (a). The other three subcases are similar.

Due to the nonuniformity in the width of the merged net, there is a need for defining *direction-dependent weight* (weight whose value depends on the direction of the edge) for edges in the VC graph of the channel. This need arises since it may be possible to place the horizontal segment of another net say  $B$  closer to the center line of the horizontal segment of the merged net on one side than on the other as shown in Fig 5.12 (b). Assignment of these weights is now discussed. If net  $B$  intersects the thin portions of the merged net (with width  $w_A$ ), and not the thick middle portion, then it can be placed at a minimum distance

of  $d_s$  above, or a minimum distance of  $d_l$  below the center line of the merged net, where

$$d_s = 0.5 * (d_{min} + w_B) \quad (5.11)$$

$$d_l = 0.5 * (d_{min} + w_B) + w_A + d_{min} \quad (5.12)$$

If net  $B$  intersects the thick middle portion of the merged net, then it has to be placed at a minimum distance of  $d_l$  above or below the merged net. Let  $x_{lB}$  and  $x_{rB}$  denote respectively the x coordinates of leftmost and rightmost pins of net  $B$ . Let  $w(B, A\bar{A})$  denote the weight of the edge  $(B, A\bar{A})$  in VC graph when directed from  $B$  to  $A\bar{A}$  where  $A\bar{A}$  denotes the merged net in the VC graph. Similarly  $w(A\bar{A}, B)$  is the weight of the same edge when directed from  $A\bar{A}$  to  $B$ . The algorithm for assigning the direction-dependent weights now follows.

**Algorithm 5 (Assignment of direction-dependent weights)**

if  $((x_{lB}, x_{rB})$  intersects  $(x_{lA}, x_{rA}))$  then  
 $d(B, A\bar{A}) = d(A\bar{A}, B) = d_l;$   
 else if  $((x_{lB}, x_{rB})$  intersects  $(x_{lA}, x_{lA})$ ) OR  $((x_{lB}, x_{rB})$  intersects  $(x_{rA}, x_{rA})$ ) then  
 $d(B, A\bar{A}) = d_s;$   
 $d(A\bar{A}, B) = d_l;$

Since it is assumed that the horizontal and vertical segments are in different layers during channel routing, implementation of the above matching algorithm is relatively simple (and which is implemented in ART) when three layers are available for interconnection, say MET1, MET2 and POLY. Then, horizontal segments can be routed in MET1, vertical segments in MET2, and the I-connectors implemented in POLY and MET1. This ensures that the I-connectors do not obstruct vertical segments in MET2 layers. Also, since the I-connectors span a small area, the extent of POLY runs will be small. However, if only two layers are available for routing, and there are pins within the thin strip of asymmetry around the axis, then the I-connectors will obstruct the vertical segments associated with those pins. In that case, the routing of those vertical segments can be completed by a maze router and this feature will be implemented in future work.



## 5.5 ART

This section presents the analog router ART which can be used for constraint-driven channel routing of analog and mixed analog/digital circuits. ART has about 17000 lines of C Code. It is built on the top of a gridless channel router ROADRUNNER (developed by Nicolas Weiner of U.C. Berkeley) by implementing the constraint-driven algorithms described in this chapter.

### 5.5.1 Program Input

ART takes as input (a) channel information (b) technology information (c) capacitance bounds and (d) symmetry information. All the input is provided in ASCII format.

The channel information consists of the pin positions and netlist for the unrouted channel. Since ART is a gridless router pins can be specified anywhere on top or bottom edges. Irregular top and bottom edges can be specified as long as the contour of each edge can be described as a piecewise constant function (i.e. a collection of parallel segments). Different nets can be assigned different widths. Nets which can be used as shield nets are marked. Layer information is necessary which provided names of the layers to be used for routing horizontal and vertical segments in the channel respectively. If three layers are available in the technology for routing, one of the layers can be specified for shielding unavoidable critical crossovers (this layer has to be between those used for horizontal and vertical segments). Moreover, the name of the layer to be used (along with the layer for horizontal segments) for short jogs in the I-connectors can also be specified.

The technology information consists of design rules and capacitance coefficients for the self and coupling capacitances of interconnects. The design rules specified are the minimum widths of nets, minimum net-to-net separation, contact overhang and contact cut size. The capacitance coefficients can be obtained automatically from the model generator CAPMOD described in Chapter 6.

The capacitance bounds are provided for the critical pairs of nets as constraints. These can be automatically generated from the performance constraints by the parasitic constraint generator PARCAR described in Chapter 4. One also has the flexibility of imposing only distance constraints on pairs of nets, in which case the router will try to keep two particular nets beyond a specified distance.

The symmetry information is provided for differential circuits by specifying the

matched pairs of nets which have to be matched, and the associated axis of symmetry.

### 5.5.2 Program Operation

A flowchart of the operation of ART is shown in Fig. 5.13. The program first merges the two nets in each matched pair and treats them as one special net. If the two nets have intersecting horizontal span the special net has nonuniform width as mentioned before. In the next stage the design rule constraints are used to assign weights to the edges in the VC graph. Different widths for different nets cause different weights for different edges<sup>2</sup>. Nonuniform widths of the special nets may cause direction-dependent weights as described in Section 5.4.2. Edges are directed from nodes representing top-edge segments to the nodes representing nets having common horizontal span with the respective top-edge segments. Similarly appropriate edges are directed for the bottom segments. Moreover, due to intersection of effective spans of pins on top and bottom edges of different nets, some of the edges have to be directed in the VC graph to avoid design-rule violation.

After the design-rule constraints are imposed, the coupling constraints are imposed on the VC graph using the algorithms presented in Section 5.3. First the crossover constraints are imposed by directing appropriate edges. Overconstraints can be detected here if the minimum crossover capacitance exceeds the imposed bound for any critical pair of nets. If three layers of interconnects are available then the unavoidable crossovers are assumed to be shielded (during post processing) to avoid overconstraints. After the crossover constraints are imposed, the coupling constraints between vertical segments are imposed by directing appropriate edges or adding directed edges. Overconstraints can again be detected if pins of critical pair of nets are so close that they cannot be shielded. After that the coupling constraints between horizontal segments are imposed, by directing appropriate edges to constrain shield nets and by increasing some edge weights if necessary to control separation between horizontal segments. After the coupling constraints are imposed, there may be some undirected edges left, which are directed to minimize channel height. Once the relative positions of all the horizontal segments are determined, their absolute positions are determined from edge weights. This also fixes the dimensions of the vertical segments connecting the pins to the horizontal segments. The segments associated with the matched pairs of nets are then separated out. The dimensions and the positions of the segments are

<sup>2</sup>an edge represents the center-line to center-line separation between two adjacent nets

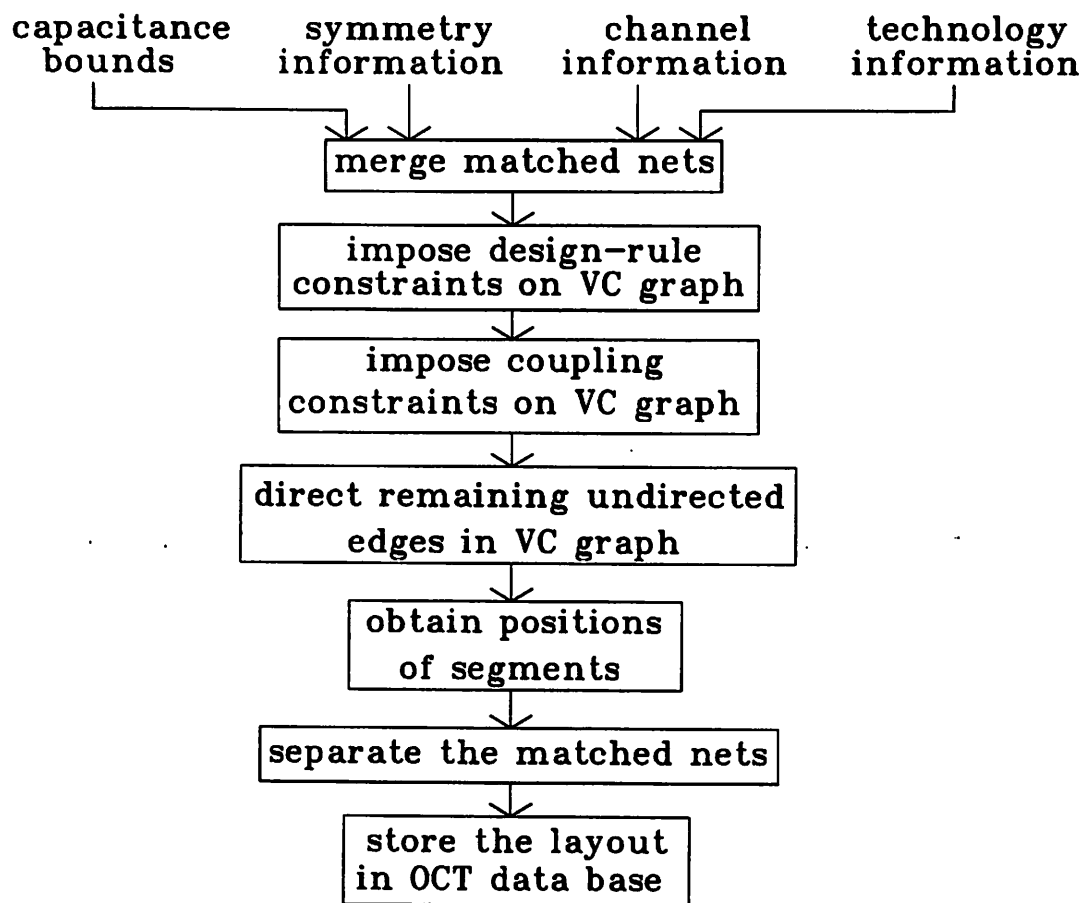


Figure 5.13: Program flow of ART

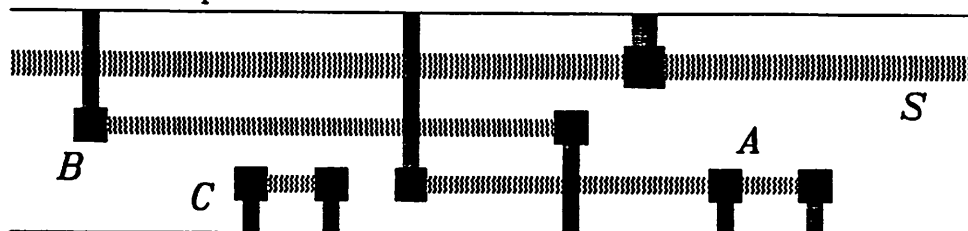
then stored in OCT data base. The routed channel can be viewed using visual editor VEM. For a more detailed description of how to use ART refer to Appendix B.

## 5.6 Results of Channel Routing

### 5.6.1 An Illustrative Example

We now present a simple example for the purpose of illustration of the ideas described in Section 5.3.2 to eliminate coupling between critical pairs of nets. A channel with four nets ( $A$ ,  $B$ ,  $C$  and  $S$ ) is considered. There is a critical coupling constraint between nets  $A$  and  $B$ , and also between nets  $A$  and  $C$ .  $S$  is a shielding net. Fig. 5.14 shows the channel routed without any constraints (i.e. with an objective of only minimizing the

channel width). The VC graph contains an undirected edge between each pair of nets having a common horizontal span.



VC graph

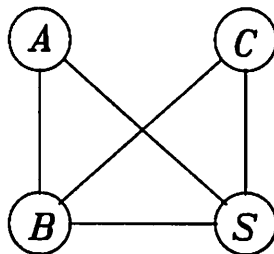
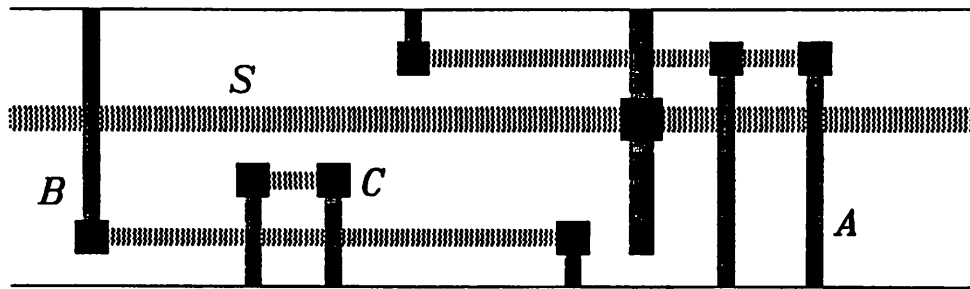


Figure 5.14: Channel Routed Without the Constraints

Fig. 5.15 shows the channel routed by imposing the constraints to eliminate coupling between  $A$  and  $B$  and also between  $A$  and  $C$ . The directed edge  $(A,B)$  prevents nets  $A$  and  $B$  from crossing. The directed edges  $(A,S)$  and  $(S,B)$  ensure that the horizontal segment of the shielding net  $S$  lies between the horizontal segments of  $A$  and  $B$ . The directed edge  $(A,C)$  is added to make sure that the adjacent parallel vertical segments of  $A$  and  $C$  do not have a common span. In addition, a vertical segment of the shielding net  $S$  is added between the adjacent vertical segments of  $A$  and  $B$  (this added intermediate segment can be made to touch the channel during post processing if design rules permit, as explained before).

### 5.6.2 SC Filter Example

The results for a fifth-order switched-capacitor filter circuit (shown in Fig. 5.16) are now presented to illustrate the imposition of bounding constraints on channel routing. The placement was generated by the program ADORE[16]. A constraint of 0.01 dB on the



VC graph

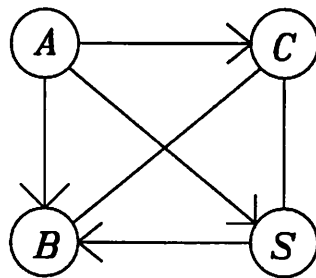


Figure 5.15: Channel Routed with the Constraints

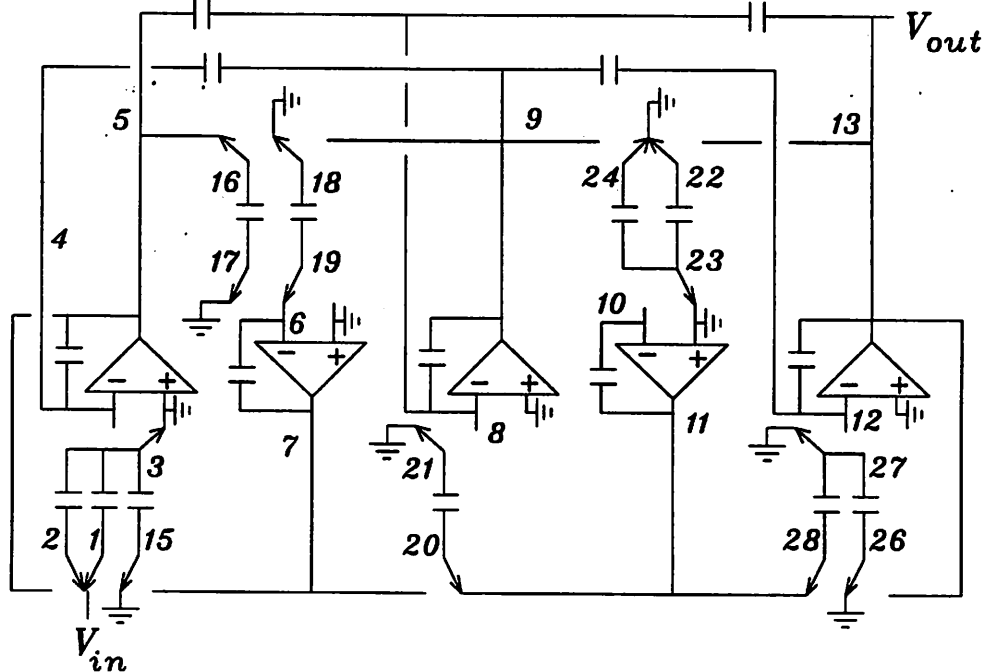


Figure 5.16: fifth-order low-pass SC filter

ripple of the frequency response was specified (this number was chosen following a procedure

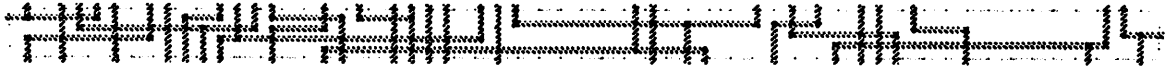
described in Section 4.8, so that it is small compared to the nominal ripple). Constraints on the critical capacitances were generated by the constraint generator PARCAR. The channels were routed by constraint-driven router ART. Three of the 30 constraints which were imposed could not be met by the router, because of the pin positions of some of the nets. But since the other constraints were met with some margins, the performance constraints were met in the first iteration. The bounding constraints and the extracted values for five of the 30 critical coupling capacitances have been shown in Table 1. The first row in Table 1 states that there is a bounding constraint of 0.0015 fF between nets 18 and 21. 21 is a sensitive net as it switches between ground and the virtual ground node 8. 18 is a large-swing net as it switches between ground and node 9 which is output of an opamp. In each of the rows of Table 1, one of the nets can be identified as a sensitive net and the other a large-swing net.

One of the channels in the layout is shown in Fig. 5.17, routed without and with the constraints. The noticeable differences resulting from the imposition of the constraints are (a) the changes in positions of the horizontal segments of some of the nets (b) introduction of a shield net. The vertical segments of the shield net are connections to shield plates used to shield the crossovers which could not be avoided by the channel router. These connections have been achieved using the constraint-driven maze router ROAD[93]. The channel height increases by about 50% due to the presence of the constraints. However, for analog circuits, the chip area is usually dominated by devices, and hence this increase is not significant. The CPU time required to route is less than 4 sec on a VAX 8650.

<i>net1</i>	<i>net2</i>	$C_{bound}$	$C_{actual}$
18	21	0.0015fF	0fF
12	28	3.3fF	1.3fF
4	15	0.01fF	0fF
27	28	0.5fF	0.01fF
23	28	0.0015fF	0.007fF

Table 1: Five of the critical net pairs, the imposed bounding constraints on their respective coupling capacitances and the actual extracted capacitances after routing

Routed without the bounding constraints:



Routed with the bounding constraints:

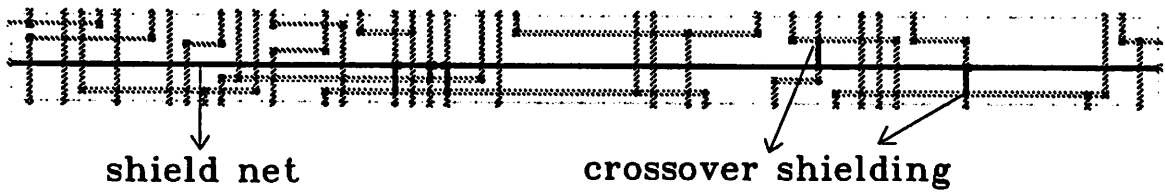


Figure 5.17: Comparison of results with and without the constraints for the SC filter

### 5.6.3 A/D Converter Example

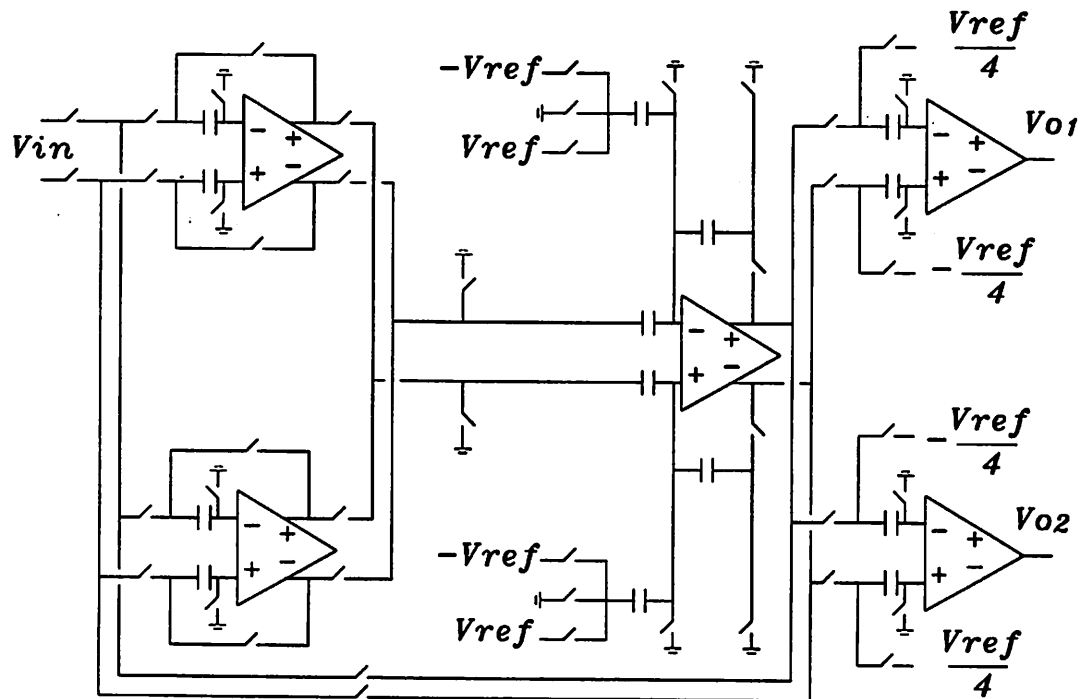


Figure 5.18: Algorithmic A/D converter

Results for an algorithmic differential A/D converter shown in Fig. 5.18 are pre-

sented to illustrate the imposition of matching constraints on channel routing. The A/D converter has two sample/hold stages at the input, followed by a precision gain stage, and two comparators at the output. One of the channels in the layout has been shown in Fig. 5.19. In this circuit, the nature of the connections and the symmetry of placement result in two pairs of nonoverlapping matched pairs in the channel. The channel routed with and without any matching constraints are shown in Fig. 5.19. The lack of mirror symmetry is evident when no matching constraints are imposed (which may cause gross mismatch in coupling capacitances as described earlier). Two I-connectors are used when matching constraints are imposed to achieve almost-perfect mirror symmetry.

Routed without the matching constraints:



Routed with the matching constraints:

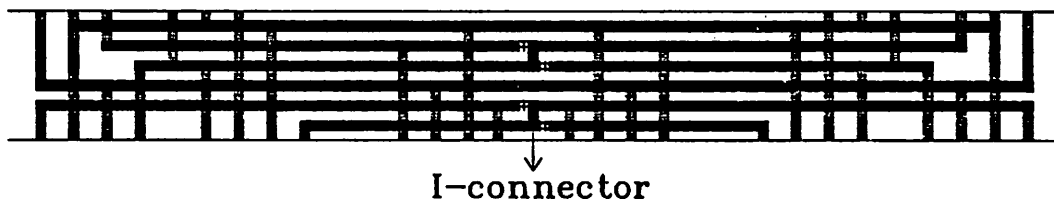


Figure 5.19: Comparison of results with and without the constraints for the A/D converter

## 5.7 Constraint-driven Area Routing

This work was performed in collaboration with Mr. Enrico Malavasi of University of Padova Italy, and hence only an overview of the work is provided here. For a more detailed description, refer to [93].

A constraint-driven framework involving the area router RoAD[29] and the constraint generator PARCAR described in Chaptersec:algo has been developed. Area routing is inherently cost-function driven as described in Section 2.4.2. In this work, a methodology



was developed in which the cost function of area routing is controlled by the performance constraints of the circuit. In the area router RoAD, weights can be assigned to parasitics (line resistances, capacitances and line-to-line capacitances). The performance sensitivities associated with the critical parasitics (detected by PARCAR) are used to control the weights assigned to the parasitics by the router. Hence, while finding a minimum-cost path the router attempts to keep the parasitics with higher weights at smaller values compared to those with lower weights. After the routing is completed, the actual values of the critical parasitics are extracted from the layout, and the associated performance degradation computed. In case some of the performance constraints are not met, the extracted parasitics are compared to the bounds on the parasitics generated by the constraint generator PARCAR. Those parasitics which exceed their bounds are considered to be responsible for violation in performance constraints, and the weights on those parasitics are increased proportional to the contribution they make towards constraint violation and the routing redone with the new weights. There is no guarantee that this iterative process will always produce a layout which meets the performance constraints. However, this process always terminates, since in each iteration through normalization, the weight of at least one critical parasitic is brought to the maximum possible value. Hence the maximum number of iterations is the number of critical parasitics. However, for practical circuits which were tried, the number of iterations was usually one or two, depending on how tight the constraints were.

## 5.8 Constraint-driven Placement

This work was carried out in collaboration with Mr. Edoardo Charbon of U.C. Berkeley and again an overview will be provided here. For a detailed description refer to [104].

Simulated annealing has been chosen as the placement algorithm as it produces good-quality layout. Although it is time-consuming, for analog circuits, number of layout components is usually small. The simulated annealing algorithm in the placer PUPPY[103] originally targeted towards digital circuits, has been modified to deal with the parasitic constraints. An additional term in the cost function is used which reflects the violation in performance constraints. Maximum and minimum estimates of net capacitances are made for a given placement (based on an estimate of net length and the per-unit-length

capacitances for maximum- and minimum- capacitance layers). As a result, the minimum and maximum possible performance degradation due to the net capacitances are made for each placement based on linearized expression using sensitivities. No cost is added if the maximum degradation is below the allowed bound. If the allowed bound is between the maximum and minimum degradations, a contribution proportional to the constraint violation is added to the cost function. When the minimum possible degradation exceeds the allowed bound placement is heavily penalized by increasing the cost function at a much faster rate with constraint violation. Analytical models generated by CAPMOD (described in Chapter 6) are used to estimate the capacitances.

## 5.9 Summary

In this chapter, a detailed description of constraint-driven channel routing and an overview of constraint-driven area routing and placement considering parasitic constraints were provided. Algorithms for mapping the constraints on a set of critical coupling capacitances into constraints in the Vertical-Constraint graph of a channel was presented. The approach involves directing undirected edges, adding directed edges and increasing the weights of the edges in the VC graph, in order to meet crossover constraints between orthogonal segments and adjacency constraints between parallel segments, while attempting to cause minimum increase in the channel height due to the constraints. Use is made of shield nets when necessary. A formal description was provided about the conditions under which the crossover and the adjacency constraints are satisfied, and were used to construct the appropriate mapping algorithms. The problem of imposing matching constraints on the routing parasitics in a channel with lateral symmetry was addressed. It was observed that perfect matching is not possible for a matched pair of nets with *intersecting* horizontal spans. A technique to achieve almost-perfect mirror symmetry in the channel was presented for such pairs of nets.

## 5.10 Appendix: Necessary condition for avoiding crossovers

In this appendix, Lemma 5.2 is proved.

**Claim:** A necessary condition for avoiding the crossover between two nets  $n_1$  and  $n_2$  is (a)  $N_t(n_2) = 0$  and  $N_b(n_1) = 0$  OR (b)  $N_t(n_1) = 0$  and  $N_b(n_2) = 0$ .

**Proof:** Suppose the necessary condition is not satisfied, i.e. (a)  $N_t(n_2) \neq 0$  or  $N_b(n_1) \neq 0$  AND (b)  $N_t(n_1) \neq 0$  or  $N_b(n_2) \neq 0$ . So, one has one of the following four possible cases.

case 1:  $N_t(n_1) \neq 0$  AND  $N_b(n_1) \neq 0$

case 2:  $N_t(n_1) \neq 0$  AND  $N_t(n_2) \neq 0$

case 4:  $N_b(n_2) \neq 0$  AND  $N_b(n_1) \neq 0$

case 3:  $N_b(n_2) \neq 0$  AND  $N_t(n_2) \neq 0$

It will be shown that in each of the above cases, it is not possible to avoid crossovers.

case1: If this case holds, then in the common horizontal span of the two nets,  $n_1$  has at least one pin on the bottom edge and at least one pin on the top edge as shown in Fig. 5.20 (a) (the pins are denoted by  $p_{n_1}$  and  $p'_{n_1}$  respectively). It is possible to draw a continuous path from  $p_{n_1}$  to  $p'_{n_1}$  through any interconnection connecting the pins of  $n_1$  inside the channel ( $n_1$  may have more than two pins in the same channel). This continuous path, since it lies entirely inside the channel, divides the channel into two halves. Also, as the two pins of  $n_1$  under consideration are within the common horizontal span of the two nets, there is at least one fixed/floating pin of  $n_2$  in each of these halves (let these two pins be denoted by  $p_{n_2}$  and  $p'_{n_2}$ ). Now it is obvious that any path from  $p_{n_2}$  to  $p'_{n_2}$  through the interconnection used for the pins of  $n_2$  in the channel will intersect the other path drawn from  $p_{n_1}$  to  $p'_{n_1}$ . Hence, the nets  $n_1$  and  $n_2$  have to cross.

case2: In this case, as shown in Fig. 5.20(b), there is at least one pin of  $n_1$  (say  $p_{n_1}$ ) and at least one pin of  $n_2$  (say  $p_{n_2}$ ) on the top edge in the common horizontal span of the two nets. Since each of these pins is in the horizontal span of the other one, there is at least one fixed/floating pin ( $p'_{n_1}$ ) of  $n_1$  to the left of  $p_{n_2}$  and at least one fixed/floating pin ( $p'_{n_2}$ ) of  $n_2$  to the right of  $p_{n_1}$  (it will be the other way if  $p_{n_1}$  was to the left of  $p_{n_2}$ ). Hence, arguing now in a fashion similar to that for case 1, it is concluded that the two nets have to cross.

case3: similar to case 1 (with  $n_1$  and  $n_2$  interchanged).

case4: similar to case 2 (with top and bottom edges interchanged).

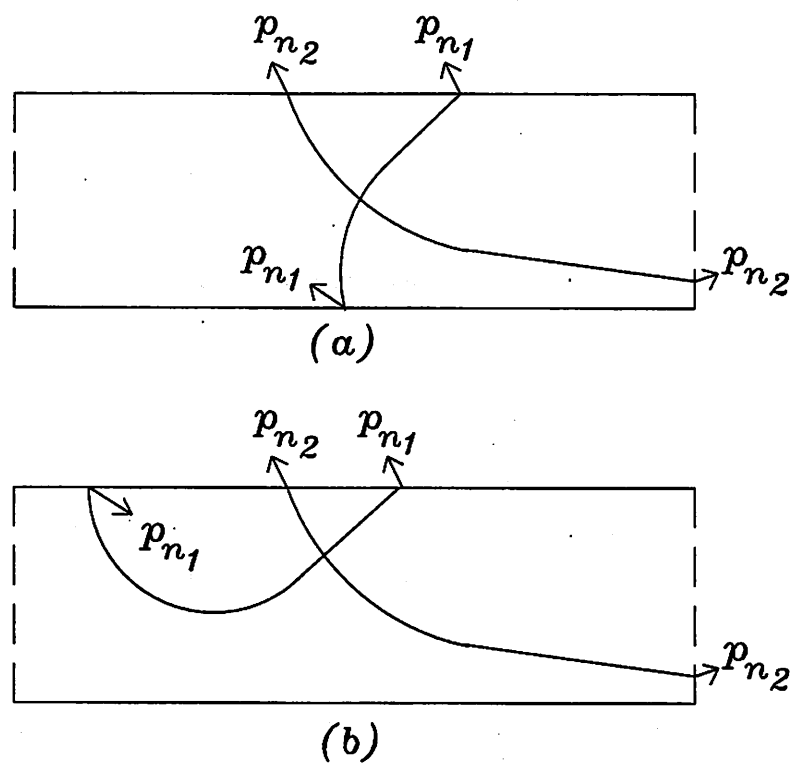


Figure 5.20: case 1 and case 2

## Chapter 6

# Analytical Models for Capacitances

In the performance-constrained approach proposed in this thesis, fast evaluation of parasitics is necessary during constraint-driven layout design while trying to meet the bounds imposed on them. Fast evaluation of parasitics is also necessary for extraction of parasitics after the layout is designed. Parasitics can be resistances, capacitances or inductances. Inductances are however not critical for chip-level design excluding very high frequencies of operation such as microwave frequencies. Resistances of conductors can be calculated with reasonable approximations using available analytical expressions. However, as mentioned in Section 2.3.2, theoretical analysis is of limited use to calculate the self and coupling capacitances associated with interconnects. This chapter proposes a modeling methodology in which the capacitance models are automatically generated using a model generator based on numerical simulations and partial knowledge of the flux components associated with the various configurations of interest.

Section 6.1 provides an overview of the approach. Section 6.2 describes the numerical technique used for computing capacitances, and Section 6.3 describes the curve-fitting technique used to approximate numerical data. Section 6.4 presents the forms chosen for analytical models for the various configurations. In section 6.5, a prototype model generator CAPMOD is described. Section 6.6 contains the results for some common layout configurations. Section 6.7 contains the summary of this chapter.

## 6.1 The Approach

Two possible practical approaches for fast estimation of capacitance in layout are (a) analytical models, and (b) table-look-up models, where the analytical or the table-look-up models are fitted to the data generated by numerical simulations or experimental measurements. For the table-look-up models, the memory requirements grow very rapidly with the increase in the number of parameters describing a given configuration, or in the range of interest for each parameter. Sophisticated interpolation techniques can be used to reduce the amount of data that needs to be stored. When the capacitance to be modeled is a smooth function of the layout parameters (true for most common configurations), compact analytical models can be fitted with the numerical or experimental data. Analytical models are quite convenient to deal with, and provide useful insight into the model dependence on various parameters. They are also quite convenient for the layout designers, when layout is synthesized manually.

Analytical models have been suggested previously for line-to-ground capacitance of a single line over a ground plane[55]. Capacitance models for a coupled pair of microstrip lines over a single layer of dielectric have been reported in the microwave literature[51]. Empirical models for capacitances have also been developed recently at TI[52], HP[53] and possibly other organizations. However, to obtain an analytical model, it is usually necessary to make certain assumptions about the process (such as the number of dielectric interfaces in the process and the range of interest for each parameter in the model). Hence, models which are developed for one type of process may not be applicable to another. Redeveloping the models manually every time for a different type of process is a tedious and time consuming task, and relies on human intuition. Hence, it is a good candidate for automation.

The approach proposed in this chapter is characterized by two key features which are: (a) *automation of model generation* and (b) *design-parameter-based modeling*.

(a) **Automation of Model Generation** : The analytical models are automatically generated based on data obtained from numerical simulations. However, one of the important aspects of obtaining an analytical model is choosing a suitable “form” for the analytical model, as it can result in a compact analytical model. For the various coupled configurations, the nature of variation of the capacitances with respect to layout parameters is often complicated, and hence choosing the suitable “form” is nontrivial. But, if the flux associated

with a configuration is decomposed into a number of components, often a *partial knowledge* of the nature of the variation of each component can be exploited to choose a suitable form.

(b) Design-Parameter-Based Modeling : The set of parameters describing a layout configuration can be categorized into two groups viz. (i) *process parameters* and (ii) *design parameters*. The process parameters have fixed nominal values for a given technology. The design parameters on the other hand can be controlled by the layout designer. As an example, for the configuration consisting of two adjacent parallel lines over a layer of dielectric, the process parameters will be the dielectric thickness, the dielectric permittivity and the conductor thickness, and the three design parameters will be the widths of the two lines, and the separation between the two lines. The choice of parameters describing a model should be dictated by its primary use. If interconnect models have to be developed for use by CAD tools in layout design and verification, often repeated evaluation of the model is desired for a few fixed configurations, but for different values of *design parameters*. For example, the coupling capacitance between two adjacent parallel lines may have to be computed for differing values of line-separation. The widths of lines in a given layout is often designed to be different from minimum width depending on the currents carried by different lines. In that case the coupling capacitances have to be computed as a function of widths of lines also. In such situations, computation involving the process parameters is a waste of CPU time as they are fixed, and hence it is more appropriate to use models including functional dependence only on the design parameters. The models have to be generated for each technology but this is done only once and automatically.

In this chapter, CAPMOD, an analytical-model generator is presented. The model generator proceeds through three phases as shown in Fig. 6.1: (a) Configuration Generation: a set of structures is generated for the configurations of interest (currently single lines, parallel lines on same layer, parallel lines on different layers and crossing lines) by varying each design parameter in the range of interest, and fixing the process parameters at the values specified for the technology; (b) Numerical Simulation: for each structure, the capacitance matrix is computed numerically by solving Laplace's equation using finite-differences; (c) Configuration-Dependent Model Fitting: suitable forms are chosen for the capacitances in the configurations of interest based on a partial knowledge of the flux components, and the analytical models are fitted to the numerically computed capacitances.

In the next section, an overview of finite-difference technique used to compute the capacitances will be provided.

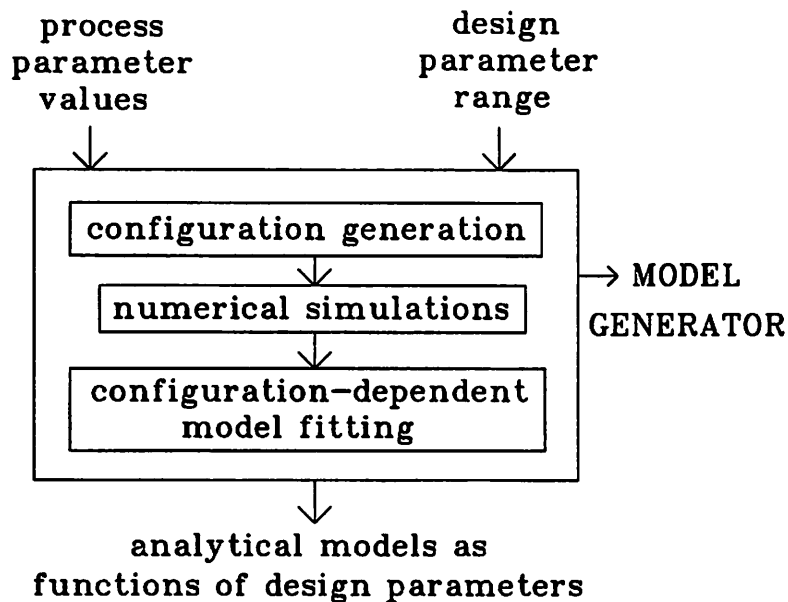


Figure 6.1: Automatic generation of analytical models

## 6.2 Numerical Simulation

Data for fitting the analytical models can be obtained either by experimental measurements, or through numerical simulations. In the proposed approach, numerical simulations are used to generate the data, as they are convenient for analyzing a very wide range of layout configurations, for different values of geometrical parameters, in a CAD environment. The accuracy of the numerical methods can be controlled by choosing the mesh sizes. Moreover, numerical simulations can be used to predict the capacitances of future technologies which do not exist yet. Numerical simulation can be performed using finite-difference, finite-element or integral equation techniques [66]. Finite-difference technique is used in the model generator CAPMOD since this scheme is relatively easy to implement and the discretization requires a mesh generation procedure that is easier to automate than in other schemes such as the finite-element method.

Finite-difference method of capacitance computation for two-dimensional case will be described here. The three dimensional case is similar. Example of a two-dimensional configuration is shown in Fig. 6.2 containing the cross section of three parallel conductors shown in black. One would like to compute the per-unit-length capacitance matrix ( i.e. per-unit-length self and coupling capacitances as described in Section 2.3.2) of such a configuration. In order to do that, successively the voltage of each conductor is set to 1v and of



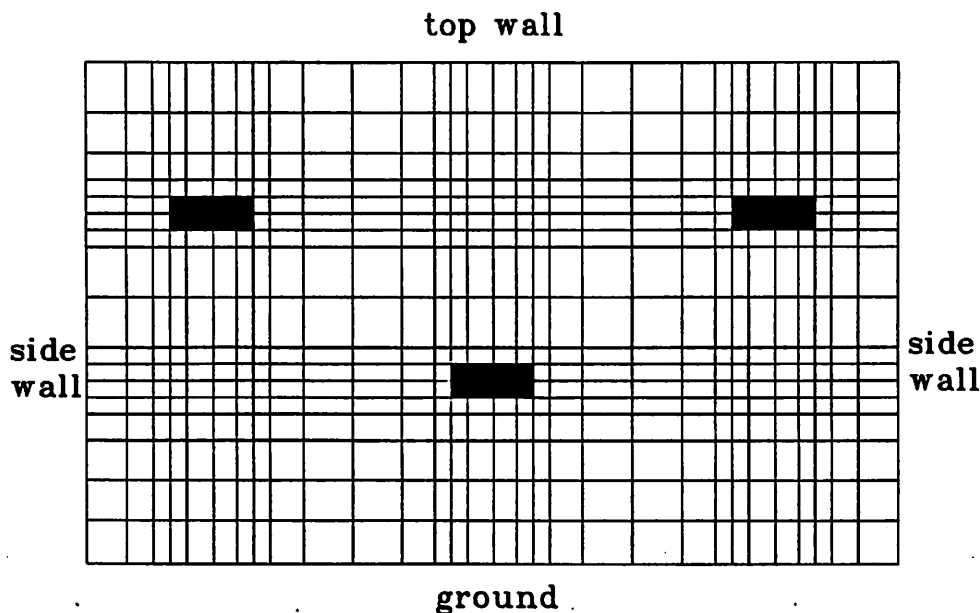


Figure 6.2: Mesh used for a two-dimensional configuration having three conductors

all the other conductors to  $0v$ . The charge on each conductor is then computed by solving for the potential distribution in the dielectric region around the conductors.

The potential  $\phi$  which is a function of two variables (say  $x$  and  $y$  as shown in Fig. 6.2) in a 2D configuration satisfies the following equation (can be derived from Maxwell's equations):

$$\partial(\epsilon \partial\phi/\partial x)/\partial x + \partial(\epsilon \partial\phi/\partial y)/\partial y = 0 \quad (6.1)$$

where  $\epsilon$  is the dielectric constant of the medium and may be in general a function of  $x$  and  $y$ . However, in regions where the dielectric is uniform (i.e  $\epsilon$  is constant) this equation can be simplified as follows (Laplace's equation).

$$\partial^2\phi/\partial x^2 + \partial^2\phi/\partial y^2 = 0 \quad (6.2)$$

Solution of Laplace's equation using finite-difference(FD) technique will now be briefly described. At dielectric interfaces  $\epsilon$  varies at least in one dimension and small modification has to be made in the solution process as mentioned later in this section. In the finite-difference method, the entire region in which the solution for the potential is sought is discretized as rectangular mesh(grid) as shown in Fig. 6.2. The purpose of the grid is to compute the potential only on a set of discrete grid points, and infer the complete solution by

interpolation. Although the grid cannot be extended to infinity, it is extended up to certain walls which are far enough to represent infinity (i.e. moving the walls further will not affect the charge distribution on the conductors). The walls are maintained at zero potential as they represent infinity. The electric field is strong near the conductors and weak far away from the conductors. As a result the potential has large gradient in regions close to the conductor, compared to regions far away from the conductors. the error involved in discretization increases for larger separation between neighboring grid points, and for larger difference in potential between neighboring grid points. Hence, the grid lines should be closely spaced near the conductors and relatively sparsely spaced in regions far away from the conductors.

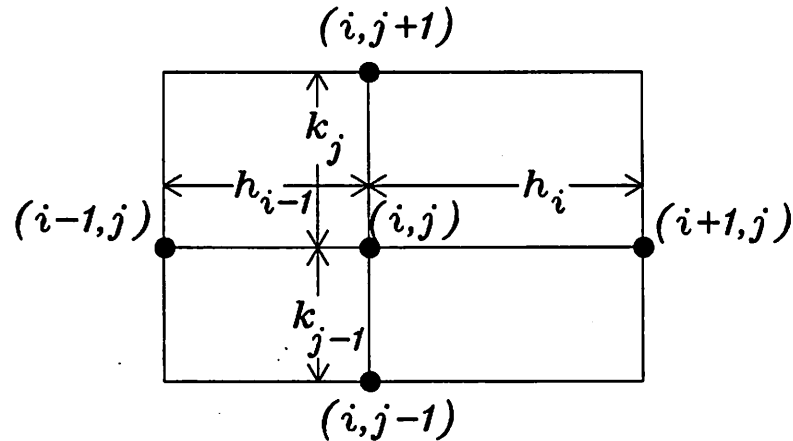


Figure 6.3: Neighboring points of  $(i, j)$  in the mesh

Let the point of intersection of the  $i^{\text{th}}$  vertical and the  $j^{\text{th}}$  horizontal grid lines be denoted by  $(i, j)$  as shown in Fig. 6.3. The separation between the  $i^{\text{th}}$  and the  $(i + 1)^{\text{th}}$  vertical grid lines is denoted by  $h_i$ , and the separation between the  $j^{\text{th}}$  and the  $(j + 1)^{\text{th}}$  horizontal grid lines is denoted by  $k_j$ . Let  $\phi[i, j]$  denote potential at point  $(i, j)$ . The finite-difference approximations to the first derivative  $\partial\phi/\partial x$  at midpoints of the segment joining  $(i, j)$  and  $(i + 1, j)$  and the segment joining  $(i, j)$  and  $(i - 1, j)$  are respectively  $(\phi[i + 1, j] - \phi[i, j])/h_i$  and  $(\phi[i, j] - \phi[i - 1, j])/h_{i-1}$ . The finite-difference approximation to the second derivative  $\partial^2\phi/\partial x^2$  is computed as the finite-difference approximation involving the two first derivatives at the two midpoints mentioned above. Since the distance between the two points where first derivative approximation are made have a separation of  $0.5(h_i +$

$h_{i-1}$ ), the approximation to the second derivative is given by:

$$( (\phi[i+1, j] - \phi[i, j])/h_i - (\phi[i, j] - \phi[i-1, j])/h_{i-1} ) / (0.5 * (h_i + h_{i-1}) ) \quad (6.3)$$

Similarly the finite-difference approximation to  $\partial^2\phi/\partial y^2$  is:

$$( (\phi[i, j+1] - \phi[i, j])/k_j - (\phi[i, j] - \phi[i, j-1])/k_{j-1} ) / (0.5 * (k_j + k_{j-1}) ) \quad (6.4)$$

Plugging these two approximations for  $\partial^2\phi/\partial x^2$  and  $\partial^2\phi/\partial y^2$  in Eq. 6.2, and rearranging terms:

$$\begin{aligned} & ( \phi[i+1, j]/h_i + \phi[i-1, j]/h_{i-1} ) / (h_i + h_{i-1}) \\ & + ( \phi[i, j+1]/k_j + \phi[i, j-1]/k_{j-1} ) / (k_j + k_{j-1}) \\ & - \phi[i, j] (1/h_i h_{i-1} + 1/k_j k_{j-1}) = 0 \end{aligned} \quad (6.5)$$

At dielectric interfaces, approximations can be made in a similar way starting from Eq. 6.1 by approximating  $\epsilon$  at the interface to be the average of  $\epsilon$  on the two sides of the interfaces. Writing Eq. 6.5 for each grid point results in a set of simultaneous equations which can be solved for the unknown potentials. Note that the potentials on the conductor surfaces are known for a given numerical simulation ( $1v$  on one of the conductors and  $0v$  for all other conductors). The potentials on the walls and the ground plane are forced to  $0v$ . In CAPMOD successive overrelaxation is used to solve the system of equations in terms of the potentials in the dielectric region. In this method, the system of equations is solved iteratively. An initial value (typically 0) is assumed for all the potentials. Then the value of each potential  $\phi[i, j]$  is updated by adding a residue which is proportional to the difference between the potential value and its value predicted from the corresponding equation (Eq. 6.5). This process is continued till the potentials converge under the desired error tolerance.

Once the potential values are solved at the grid points, finite-difference approximation is used to compute the electric field (which is the derivative of the electric potential). The charge density at any point on a conductor is the product of the dielectric constant and the electric field. Hence, once the charge density is computed at the grid points lying on the surface of the conductors, the total charge on each conductor can be computed by integrating the charge density over the conductor surface. As described in Section 2.3.2, after the charges are computed the various elements of the capacitance matrix can be computed.

## 6.3 Model Fitting

For each configuration of interest, the model generator models each capacitance of the capacitance matrix as a function of the design parameters  $p_i, i = 1, \dots, N_p$ . There are certain advantages in choosing a polynomial form for the function describing the model. Polynomials are easy to compute and they allow to tradeoff accuracy and complexity of a model by choosing the appropriate degree for the polynomial approximation.

The model needs to be valid only in a finite range of values for the parameters. If the dependence of the model on each parameter is sufficiently smooth (which is true for the common configurations of interest), a polynomial can always be chosen to fit a finite set of numerical data within any given error limit. A straightforward fitting may often result in a polynomial with a very high degree. An understanding of the flux components associated with a given configuration can be used to choose a suitable *form* for the polynomial. This will be illustrated in Section 6.5 for the configurations handled by the model generator.

### 6.3.1 Choosing a form for the polynomial

Choosing a form of the polynomial involves choosing a *suitable transformation* of the parameters and *fixing* the values of some of the *degrees* and *coefficients* in the polynomial. For example, let the capacitance of a structure depend on two design parameters  $x_1$  and  $x_2$ . If the functional dependence is smooth, in general the capacitance can be modeled as a polynomial of degree  $n_1$  in  $x_1$  and degree  $n_2$  in  $x_2$ . However, suppose for this example, it is known that the flux associated with the configuration has two components which are respectively functions of  $x_1$  and  $x_2$ , and the first component is proportional to  $x_1$ . The exact nature of variation for the second component is not known, but it is known that it has a singularity at  $x_2 = 0$ , and asymptotically approaches a constant as  $x_2$  tends to infinity. Then, a suitable form for the capacitance is  $k_0x_1 + k_1 + k_2/x_2 + k_3/(x_2)^2 + \dots + k_{n+1}/(x_2)^n$ .

Here, a transformation of the second variable is done by taking its reciprocal. The degree  $n_1$  of  $x_1$  is fixed at 1. The coefficients of all the cross terms between the two variables are set to zero, i.e. the total flux is decoupled into two components. In the resulting form of the polynomial, the coefficient of  $x_1$ , the constant  $k_1$ , the highest power of  $1/x_2$  and the coefficients of powers of  $1/x_2$  are the variables to be determined.

### 6.3.2 Computation of coefficients of the polynomial

Let it be assumed that the highest power of each parameter is fixed. Then the problem is to determine the coefficients of a polynomial with a fixed number of terms. How these degrees can be increased automatically from the lowest possible values in an iterative fashion (till the desired accuracy is achieved) will be described later.

A curve-fitting technique described in [54] is followed for obtaining the coefficients. The procedure is briefly discussed here for completeness. Let the capacitance be denoted by  $C$ , and the transformed design parameters be denoted by  $p_i$ ,  $i = 1, \dots, N_p$ . Let  $N_t$  be the number of terms in the chosen form of the polynomial expression for  $C$ .

$$C = \sum_{j=1}^{N_t} a_j t_j(p_1, \dots, p_{N_p}) \quad (6.6)$$

where  $t_j(p_1, \dots, p_{N_p})$  is the  $j$ th term in the polynomial (contains product of powers of  $p_i$ ), and  $a_j$  is its coefficient. The goal is to determine the set of coefficients  $\{a_j\}$  which minimizes the error defined with respect to the data points. The error criterion used is the *maximum relative error* denoted by  $e_{r\_max}$ . This is because, in practice, it is usually desirable to obtain a model which has some prescribed *percentage* accuracy.

The set of data points is denoted by  $D = \{D_k, k = 1, \dots, N_d\}$ , where each  $D_k$  in turn is a set of values for the transformed design parameters  $\{p_i\}$ . The data points are generated by varying the design parameters in the range of interest as mentioned earlier. The value of the capacitance obtained from numerical solution at data point  $D_k$  is denoted by  $v(D_k)$ , and the polynomial expression of the model evaluated at the same data point by  $m(D_k)$ .  $m(D_k)$  is a function of  $\{a_j\}$  only. Then the problem of *minimizing the maximum relative error* can be represented as

$$\text{minimize } e_{r\_max} \quad (6.7)$$

such that

$$|m(D_k) - v(D_k)|/|v(D_k)| \leq e_{r\_max} \quad (6.8)$$

Expanding the absolute value of the difference, one has the following formulation

$$\text{minimize } e_{r\_max} \quad (6.9)$$

such that

$$m(D_k) - v(D_k) - |v(D_k)|e_{r\_max} \leq 0 \quad k = 1, \dots, N_d \quad (6.10)$$

$$v(D_k) - m(D_k) - |v(D_k)|e_{r\_max} \leq 0 \quad k = 1, \dots, N_d \quad (6.11)$$

Since  $m(D_k)$  is a linear function of  $\{a_j\}$ , this can be considered as a linear programming problem in  $\{a_j\}$  and  $e_{r\_max}$ . The solution of the problem gives the coefficients which minimize the error, and the minimum error itself.

### 6.3.3 Computation of highest powers of parameters

The highest powers of some of the parameters can be treated as variables as mentioned earlier. For each of these parameters, the degree can be raised from zero upward till the desired accuracy is obtained. At any point in this process, the degree of that parameter can be incremented which causes maximum decrease in error for unit increase in complexity of the polynomial. The complexity of the polynomial is defined as the sum of the powers to which various parameters are raised in each term of the polynomial.

## 6.4 Forms chosen for the configurations

Currently, the model generator considers the following configurations : (a) a single line, (b) a pair of crossing lines (c) a pair of parallel lines on the same layer, and (d) a pair of parallel lines on different layers. For the 3D crossover configuration (crossover), total *lumped capacitances* are modeled. For the 2D configurations ((a),(c) and (d)), only the *per-unit-length* capacitances are modeled, since these capacitances can be used to compute the lumped capacitances of lines after their lengths are estimated. They can also be used along with other quantities such as per-unit-length resistances and inductances for simulating single or coupled transmission lines.

In a coupled configuration, the capacitance to ground of each line is less than that of an isolated line. A way to model this effect is to define a new per-unit-length capacitance to ground of each line for the coupled configuration. However, when a line is part of a 3D coupled configuration such as a crossover, the per-unit-length capacitance of each line varies in a complicated fashion along its length. To overcome this problem, a *lumped correction*

*capacitance* is modeled for each line in the 3D configuration which has to be subtracted from the total estimate of its capacitance to ground. Similarly, for each line in a 2D configuration, a *per-unit-length correction capacitance* is modeled.

The forms for the capacitances are chosen based on decomposition of flux components for the various configurations, as described in detail later. For several configurations, the nature of variation of a flux component with layout parameters may not be known exactly, but it may be known that a given component monotonically increases (or decreases) with a certain layout parameter  $x$  for small  $x$ , and asymptotically approaches a constant as  $x$  becomes large. In this case, if a polynomial in  $x$  is used to model this saturating relationship, then a very high-degree polynomial may be required. However, if a polynomial in  $(1/x)$  is used, then a much more compact model can be obtained as the constant in the polynomial represents the asymptotic value of the model, and the powers of  $(1/x)$  model the variation for small values of  $x$ . This works if the flux component actually has a singularity at  $x = 0$ . If that is not so, then a polynomial in  $1/(x + x_0)$  is used where  $x_0$  is a constant chosen to minimize the error.

The notations used for modeling are:

- $C_0$  : Capacitance to ground of a single isolated line,
- $C_{12}$  : Coupling Capacitance between line1 and line2,
- $C_{1c}$  : Correction Capacitance associated with line1,
- $C_{2c}$  : Correction Capacitance associated with line2,
- $F_p$  : Parallel-Plate Flux,
- $F_f$  : Fringe Flux,
- $\mathcal{P}(p)$  : A polynomial in a variable  $p$ .

$k_0, k_1, k_2, \dots$  are the coefficients in the various analytical forms. These coefficients are computed by the model generator based on the numerical data using the curve-fitting technique presented in Section 6.3. It is to be kept in mind that the same constant (say  $k_2$ ) can have different values and dimensions in different expressions.

#### 6.4.1 Single-Line Configuration

The design parameter for the single-line configuration shown in Fig. 6.4 is the width of the line. There may be dielectric interfaces below or above the conductor. The total flux associated with the conductor has a parallel-plate component  $F_p$  and a fringe

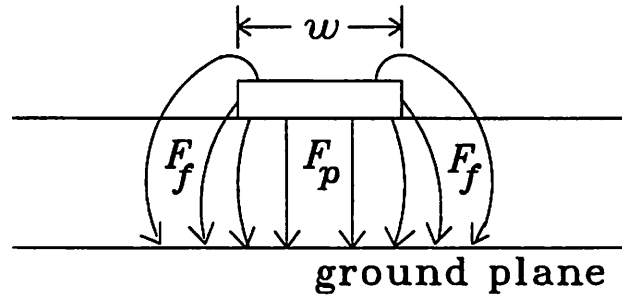


Figure 6.4: Flux components of single-line configuration

component  $F_f$  due to excess charge at the edges of both top and bottom plate and on the side walls. The parallel-plate component is proportional to  $w$ . The fringe component is insensitive to  $w$  unless the width of the conductor is very small compared to its distance from the ground plane, in which case it decreases with  $w$  due to the interaction of edge charges on the top and bottom plates. The fringe component is assumed to be a constant independent of  $w$ , with the understanding that a polynomial in  $1/(w + w_0)$  can be used to model the nonlinearity for small widths, where  $w_0$  is a constant to be determined. With these assumptions, the capacitance is modeled by the relationship:

$$C_0 = k_0 + k_1 w$$

where  $k_0$  and  $k_1$  are computed from numerical data, and whose values are going to be different for different layers. The above capacitance is only the per-unit-length capacitance of a line. For lines of finite length, excess components have to be added at both ends.

#### 6.4.2 Crossover Configuration

For the crossover configuration, shown in Fig. 6.5, the design parameters are the width  $w_1$  of the bottom line (line1) and width  $w_2$  of the top line (line2). The coupling capacitance between two lines is defined as the ratio of mutual flux (the flux originating from one line and terminating on the other line) and the voltage difference maintained between the two lines. This flux is decomposed into three components (a) a parallel-plate component  $F_p$ , (b) a fringe component  $F_{f1}$  associated with edges of line1, and (c) a fringe component  $F_{f2}$  associated with edges of line2 as shown in Fig. 6.6. The component  $F_p$  is proportional to the overlap area  $w_1 w_2$ . If the width of the second line were infinity, then  $F_{f1}$  would have been uniformly distributed along the edges of line1. However, due to the



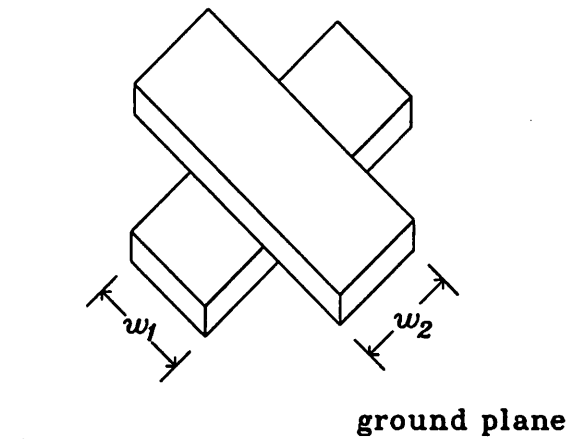


Figure 6.5: Crossover configuration

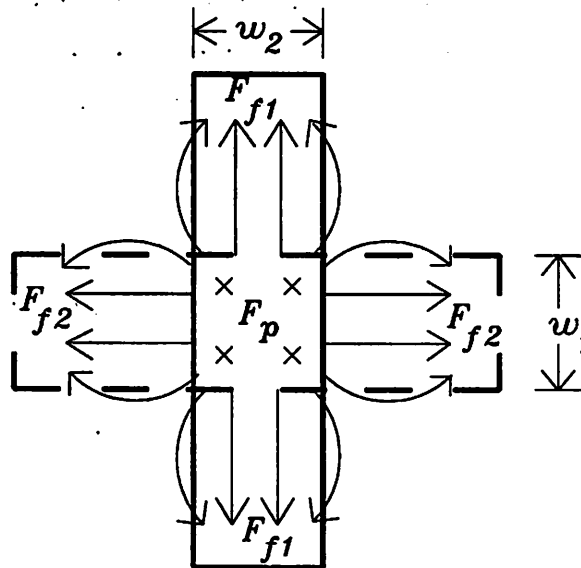


Figure 6.6: Components of mutual flux in a crossover

presence of edges of line2,  $F_{f1}$  is further split into two subcomponents; one is proportional to the width  $w_2$ , and the other is a constant representing the excess contribution to  $F_{f1}$  because of the edges of line2. Similarly  $F_{f2}$  is broken into a subcomponent proportional to  $w_1$ , and a constant. Hence, the coupling capacitance which is proportional to the sum of the three flux components is modeled in a *bilinear form* as follows.

$$C_{12} = k_0 + k_1 w_1 + k_2 w_2 + k_3 w_1 w_2$$

The first three terms in the above expression are due to fringe effect. Because of the constant  $k_0$ , and because  $k_1$  and  $k_2$  can be different (mainly because the thicknesses of the two lines may be different), the fringe component is not simply proportional to the perimeter of the overlap area as often believed.

Now let the correction capacitance of line1(bottom line) due to the presence of line2(top line) be considered. The correction capacitance of line1 is proportional to the flux of line1 intercepted by line2, that is the amount of flux originating from line1, which would have terminated on the ground plane if line2 were not present. The flux of line1 intercepted by line2 is mostly a part of the flux associated with the top plate of line1, since line2 is located on the top of line1. The flux originating from top plate of an isolated line and terminating on the ground plane consists of two fringe components associated with the two edges. When a second line is present on the top of the line, the intercepted flux can be modeled as a sum of a constant component and another which is proportional to the width  $w_2$  of the top line (using an argument similar to that of  $F_{f1}$ ). Hence, the correction capacitance of bottom line is modeled as:

$$C_{1c} = k_0 + k_1 w_2$$

The flux of the top line intercepted by the bottom line has two components (a) a parallel-plate component proportional to  $w_1 w_2$  and (b) a fringe component which has a subcomponent proportional to the width of bottom line and an excess component (using an argument similar to that of  $F_{f1}$ ). Hence, the correction capacitance is assigned the form:

$$C_{2c} = k_0 + k_1 w_1 + k_2 w_1 w_2$$

### 6.4.3 Parallel Lines on the Same Layer

For the parallel-line configuration shown in Fig. 6.7, there are three design parameters: the line-widths  $w_1$ ,  $w_2$ , and the line-separation  $s$ . The mutual flux between the two lines has a parallel-plate component  $F_p$  and a fringe component  $F_f$ .  $F_p$  is proportional to  $1/s$ . On the other hand,  $F_f$  is a very complicated function of the separation and the widths of the lines. Obviously,  $F_f$  asymptotically goes to zero as separation goes to infinity. Moreover,  $F_f$  goes to infinity as  $s$  goes to zero. This can be appreciated by noting that the coupling capacitance between two zero-thickness strips is solely due to fringe flux and it goes to infinity as  $s$  goes to 0 (for any finite line-widths). Hence the dependence of  $F_f$  on  $s$  is modeled as a polynomial in  $1/s$ .

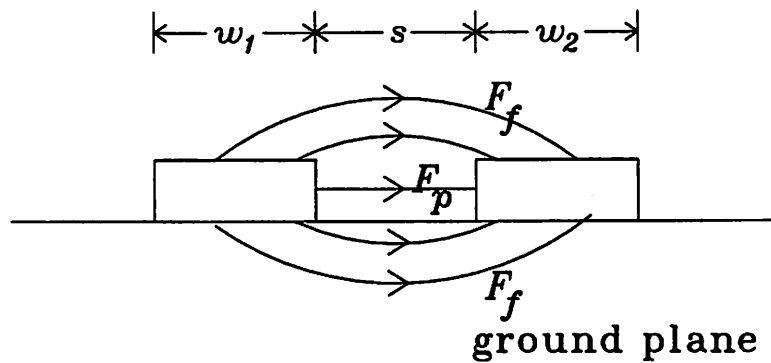


Figure 6.7: Parallel lines on the same layer

$F_f$  also has a relatively weak dependence on widths of the lines. For fixed separation,  $F_f$  increases with the width of each line and gradually becomes insensitive to further increase in line width. This can be attributed to the fact that the charge density induced on top and bottom plates of each line due to a finite potential on the other line falls (and asymptotically goes to zero) with increasing distance from the other line. Hence, when the width of the line is increased, after certain point, further increase in width of the line has no effect on its total induced charge.  $(s + w_1)$  is the distance of the edge of line1 farther from line2 from line2. When this distance approaches infinity the influence of the width of line1 on the fringe component  $F_f$  vanishes. Hence, influences of widths of line1 and line2 on the fringe component  $F_f$  are respectively modeled by adding polynomial in  $1/(s + w_1)$  and in  $1/(s + w_2)$  to the polynomial in  $1/s$ . Since the parallel-plate component can be absorbed into the polynomial in  $1/s$ , the form for the coupling capacitance is chosen as:

$$C_{12} = \mathcal{P}(1/s) + \mathcal{P}(1/(s + w_1)) + \mathcal{P}(1/(s + w_2))$$

Now the correction capacitances are considered. Part of the *fringe* flux of each line is intercepted by the other line. This intercepted flux again goes to 0 as  $s$  tends to infinity. But unlike the mutual flux, it does not approach infinity as  $s$  tends to 0 (by definition, the intercepted flux can not be more than the flux associated with the line in the absence of the other line). Hence, the dependence of intercepted flux on  $s$  is modeled as a polynomial in  $1/(s + s_0)$ ,  $s_0$  being a constant to be determined. To choose a form for dependence on widths, one should note that the fringe flux of each line associated with only the edge facing the other line is significantly affected. Thus, the correction capacitance of line1 is not sensitive to its own width  $w_1$ , and vice versa for line2. Following a similar argument as

that for the coupling capacitance, the dependence of correction capacitances  $C_{1c}$  and  $C_{2c}$  on line widths are modeled by polynomials in  $1/(s + s_0 + w_2)$  and  $1/(s + s_0 + w_1)$  respectively. Hence, the forms for the correction capacitances are:

$$C_{1c} = \mathcal{P}(1/(s + s_0)) + \mathcal{P}(1/(s + s_0 + w_2))$$

$$C_{2c} = \mathcal{P}(1/s + s_0) + \mathcal{P}(1/(s + s_0 + w_1))$$

#### 6.4.4 Parallel Lines on Different Layers

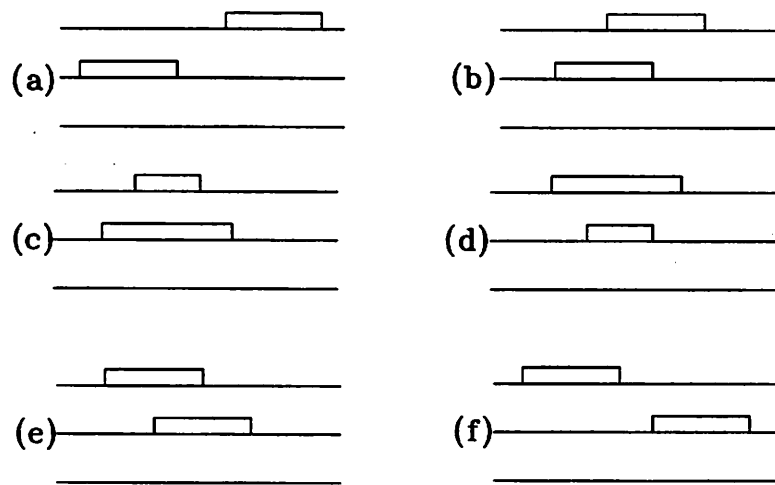


Figure 6.8: Parallel lines on different layers

Two parallel lines on different layers can be in one of the six possible topologically different configurations shown in Fig. 6.8. However, from symmetry considerations, models for configurations (e) and (f) can be obtained from those of (b) and (a) respectively. Hence, only configurations (a),(b),(c) and (d) need to be considered. Configuration (a) corresponds to nonoverlapping parallel lines and configurations (b),(c) and (d) correspond to overlapping parallel lines.

##### Nonoverlapping Parallel Lines

In this configuration ((a)) there is no parallel-plate mutual flux between the two lines and hence coupling capacitance is solely due to fringe flux  $F_f$ (Fig. 6.9). However, the nature of variation of the mutual flux with the design parameters is very similar to that of parallel lines on the same layer (Section 6.4.3). The only significant difference is that the

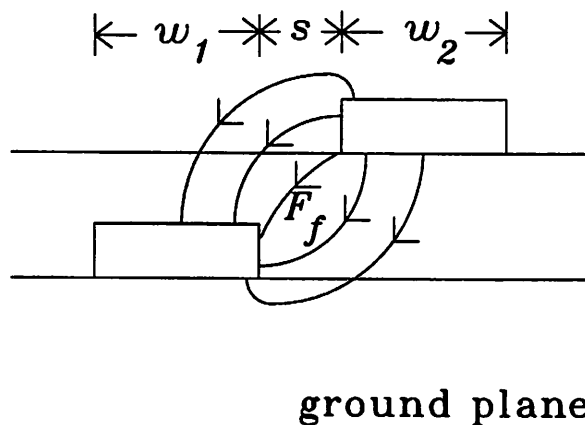


Figure 6.9: Components of mutual flux for nonoverlapping lines

mutual flux for this configuration is finite at  $s = 0$  instead of having a singularity. Hence, the forms for the coupling and the correction capacitances are chosen as follows:

$$C_{12} = \mathcal{P}(1/(s + s_0)) + \mathcal{P}(1/(s + s_0 + w_1)) + \mathcal{P}(1/(s + s_0 + w_2))$$

$$C_{1c} = \mathcal{P}(1/(s + s_0)) + \mathcal{P}(1/(s + s_0 + w_2))$$

$$C_{2c} = \mathcal{P}(1/(s + s_0)) + \mathcal{P}(1/(s + s_0 + w_1))$$

### Overlapping Parallel Lines

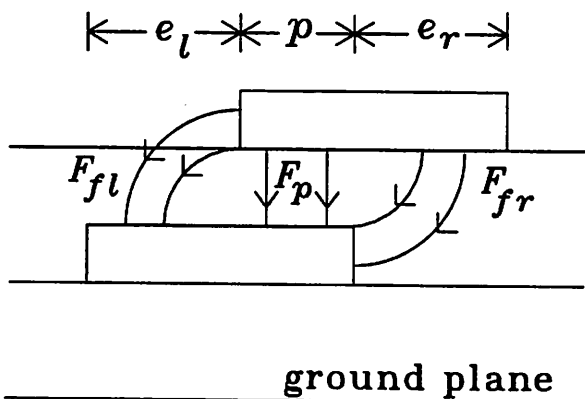


Figure 6.10: Components of mutual flux for overlapping lines

In each of the configurations (b), (c) and (d) there is a parallel-plate component as well as a fringe component of the mutual flux. Let the original set of design parameters  $\{s, w_1, w_2\}$  be transformed into a set  $\{p, e_l, e_r\}$  where  $p$  is the overlap width, and  $e_l$  and  $e_r$

are respectively the extensions of the conductors on the left and right of the overlap width. This is illustrated only for configuration (b) in Fig. 6.10. The parallel-plate component of flux is proportional to  $p$ . The fringe flux has two subcomponents associated with  $e_l$  and  $e_r$  respectively. The one associated with  $e_l$  has a finite value for  $e_l = 0$  and increases with  $e_l$  until it becomes insensitive to further increase in  $e_l$ . This can be explained in a manner similar to that used for explaining the variation of fringe component with respect to line width for parallel lines on same layer. Hence, using the strategy described before for this type of variation, it is modeled as a polynomial in  $(1/(e_l + e_{l0}))$ . The fringe flux on the right is similarly modeled as a polynomial in  $(1/(e_r + e_{r0}))$ .  $e_{l0}$  and  $e_{r0}$  are the constants to be determined. Hence, the form of the coupling capacitance is chosen as:

$$C_{12} = k_0 p + \mathcal{P}(1/(e_l + e_{l0})) + \mathcal{P}(1/(e_r + e_{r0}))$$

Now, the correction capacitances for configuration (b) is considered. Due to the presence of the top line, the fringe flux of the bottom line (mainly that associated with the top plate) is partly intercepted. For fixed values of  $p$  and  $e_l$ , when  $e_r$  increases from 0, the intercepted flux increases and gradually approaches a constant just like the mutual flux. However, for fixed values of  $p$  and  $e_r$ , when  $e_l$  increases from 0, the intercepted flux decreases and approaches a constant, because when the top line is beyond a certain distance from the left edge of the bottom line, the left fringe flux associated with the first line is no longer affected by the top line. For fixed  $e_l$  and  $e_r$ , the intercepted flux of bottom line is insensitive to  $p$  for large  $p$ , as expected, but decreases to some extent for small  $p$  (similar to the dependence on  $w_1$  for crossover). The intercepted flux associated with the top line has just the opposite kind of variations with  $e_l$  and  $e_r$  (increases with  $e_l$  and decreases with  $e_r$ ), but a parallel-plate component of flux of the top line (proportional to  $p$ ) is intercepted by the bottom line. Hence the forms for the correction capacitances for configuration (b) are chosen as

$$C_{1c} = \mathcal{P}(1/(p + p_0)) + \mathcal{P}(1/(e_l + e_{l0})) + \mathcal{P}(1/(e_r + e_{r0}))$$

$$C_{2c} = k_0 p + \mathcal{P}(1/(e_l + e_{l0})) + \mathcal{P}(1/(e_r + e_{r0}))$$

The same form is chosen for configurations (c) and (d), because the two correction capacitances have similar variation with respect to the three variables for configurations (c) and (d) (although, whether a correction capacitance will increase or decrease with the variables  $e_l$  and  $e_r$  depends on the configuration).

## 6.5 CAPMOD

### 6.5.1 Program Input

The program takes the values of process parameters for the technology of interest and the range of interest of the design parameters. The finite-difference numerical simulator can handle a process with arbitrary levels of interconnects as shown in Fig. 6.11.

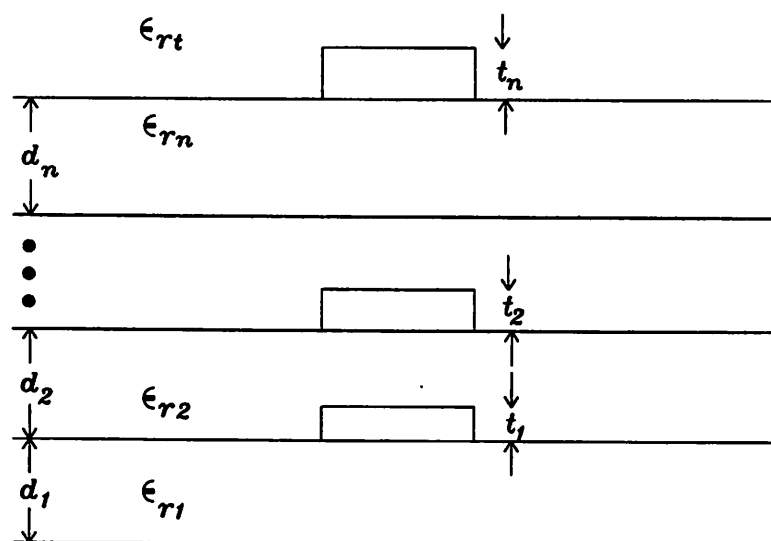


Figure 6.11: Process parameters of a process with arbitrary levels of interconnects and dielectric interfaces

The process parameters for the process shown in Fig. 6.11 are (a)  $n$ : the number of layers (b)  $d_1, d_2, \dots, d_n$ : thicknesses of the layers. (c)  $\epsilon_{r1}, \epsilon_{r2} \dots \epsilon_{rn}$ : relative permittivities of the layers. (d)  $\epsilon_{rt}$ : relative permittivity of the medium which extends up to infinity above the topmost layer (air for real processes). (e)  $t_1, t_2, \dots, t_n$ : thicknesses of the conductors in the respective layers.

The model generator however generates analytical models only for  $n = 3$ , although it can be easily extended to handle  $n$  larger than 3. Two levels of interconnects are assumed with the third layer extending up to a certain height beyond the top of the second level of interconnect. Hence the process parameters fed to the program are  $d_1, d_2, d_3, \epsilon_{r1}, \epsilon_{r2}, \epsilon_{r3}, \epsilon_{rt}, t_1$  and  $t_2$ .

Besides the process parameters, the program also takes the minimum and the

maximum values for the design parameters which are the line widths and line separations. The minimum line widths and minimum separation between adjacent lines on two layers are decided by the technology. The maximum width on each layer has to be specified based on the maximum line width which can possibly be used for design. For separation between parallel lines, the maximum value on each layer should be chosen so that, beyond this distance, the ratio of the coupling capacitance between two adjacent parallel lines to the self capacitance of each line is always below a specified threshold, say 1% (this ratio has the physical interpretation of being the ratio of flux coupling per unit voltage difference between the two lines, to the flux between any line and the ground plane per unit voltage difference, and hence can be considered to be a measure of coupling intensity). The layout resolution, i.e. the minimum dimension which all the layout dimensions should be multiples of, has also to be specified.

### 6.5.2 Program Operation

CAPMOD is about 7000 lines of c-code. The flow diagram of the program is shown in Fig. 6.12.

First, the program generates a series of structures for (a) single lines (b) parallel lines (on same layer and on different layers) and (c) crossing lines, by varying the design parameters in the range of interest with an increment determined by the layout resolution of the technology. Model generation can be carried out only for selected configurations by issuing proper options.

For each of the structures generated, the side walls and the top wall (as shown in Fig. 6.2) are generated automatically to represent infinity. Let  $h_g$  denote the height of a conductor from the ground plane. From numerical simulations it has been observed that if the side walls are at a distance of about  $3h_g$  from the side edges of the conductor, and the top wall at a distance of about  $8h_g$  from the top edge of the conductor, then a further movement of the walls does not have any significant effect on the conductor charge. Hence the walls at these distances are created to represent infinity. In case of several conductors, the walls are put at these distances from the nearest edge of the nearest conductor.

After the walls are generated, the mesh is created automatically. The mesh generation will be explained only for the two dimensional configurations with the aid of Fig.



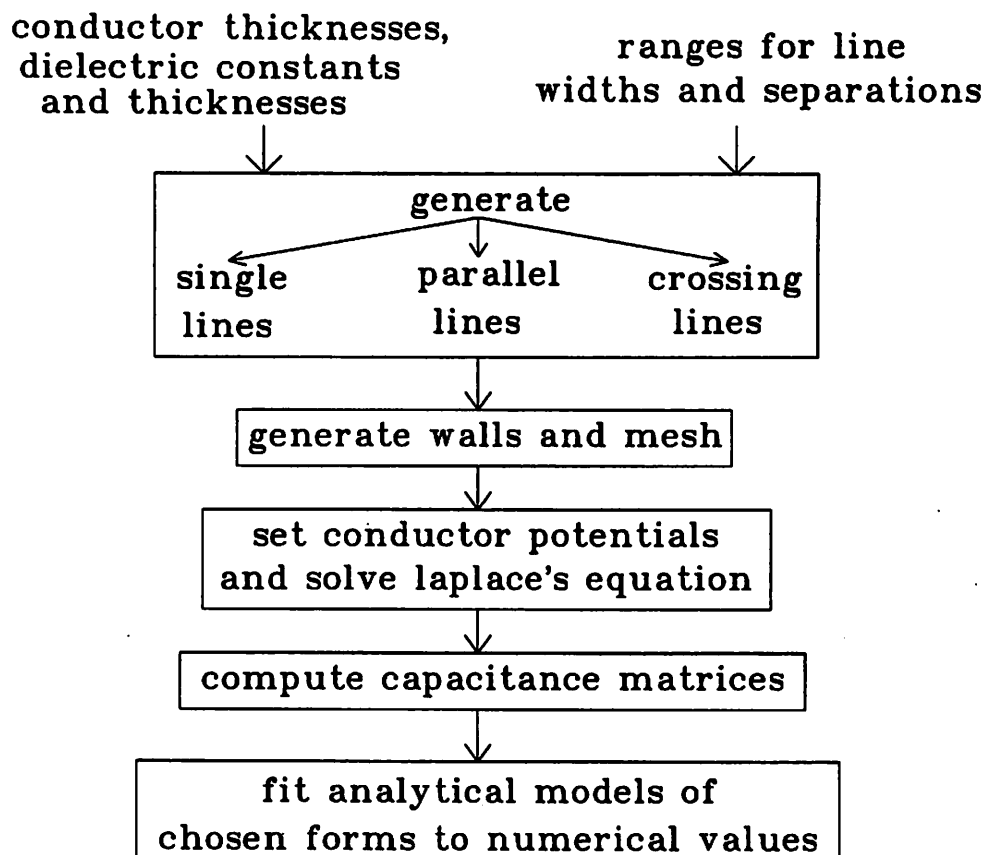


Figure 6.12: Flow diagram of CAPMOD

6.13. The three-dimensional case is similar. The generation of vertical grid lines will now be explained. The electric field in the  $x$  direction increases as one moves towards the vertical edge of a conductor and hence the vertical grid lines are most closely spaced near the vertical edges. To be more precise, if  $x_m$  is the minimum of all the separations between two neighboring vertical edges, then the minimum spacing between the grid lines is made equal to  $fx_m$ , where  $0 < f < 1$ . The separation between the grid lines is then expanded by a factor  $r$  ( $r > 1$ ) away from each vertical edge, until a side wall is hit, or the grid line crosses halfway between two neighboring vertical edges. The smaller the values of  $f$  and  $r$ , the less will be the error in computation, but at the expense of CPU time (since more grid lines will be used). Default values of factors  $f$  and  $r$  are used in the program, although, they can be controlled to control the accuracy. The horizontal grid lines are formed in a similar way with respect to the horizontal edges which include the ground plane.

After the mesh is generated, the voltage of one of the conductors is set to  $1v$  and

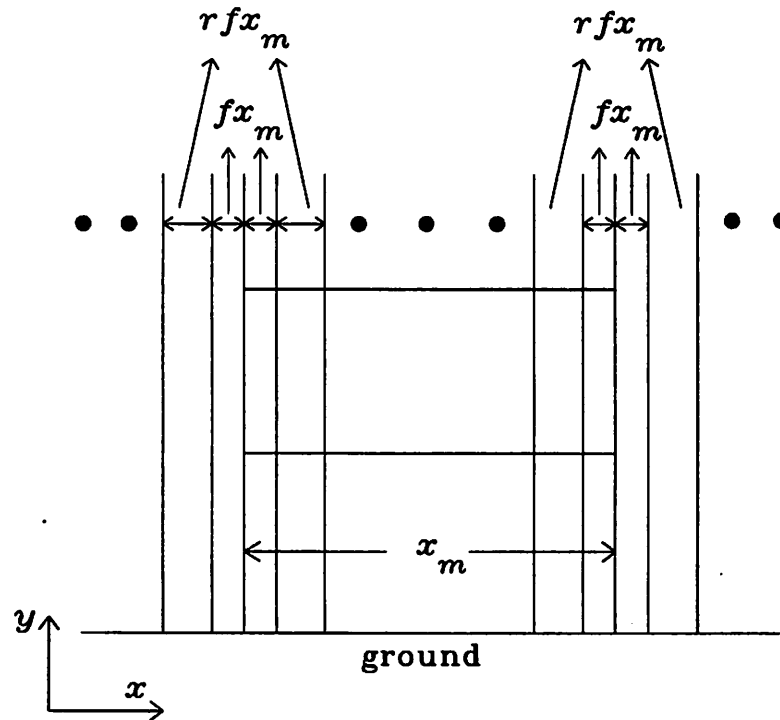


Figure 6.13: Flow diagram of CAPMOD

voltages of all other conductors are set to  $0v$ . Then the potentials of all the grid points are computed iteratively by solving the discretized version of Laplace's equation and using successive overrelaxation for the system of equations as described in Section 6.2. The self capacitance of the conductor whose voltage is set to  $1v$  and its coupling capacitance with respect to each of the other conductors are computed by calculating the charges on the conductors. 2D or 3D analysis is performed depending on the configuration. The analytical model of chosen forms (described in Section 6.4) are then fit into the numerical data (as explained in Section 6.3).

As mentioned earlier the data points are chosen by incrementing the design parameters by the specified value of layout resolution. Larger the set of data points, more will be the accuracy of estimation, but at the expense of CPU time. Hence, speed-accuracy tradeoff can be achieved by varying the specified value of layout resolution. However, there are more sophisticated techniques of automatically choosing data points based on some statistical error criterion. For a description of these methods refer to [97]-[102].

One limitation of the framework is that the internal numerical simulator being used in the model generator can currently handle only Manhattan geometries. The conductors

have rectangular cross section, and the dielectric layers are stratified (interfaces are planes parallel to each other). This assumption is made due to the limitations of the experimental finite-difference field simulator currently being used. Sophisticated field-simulation packages have been reported [58], which are capable of analyzing more complicated configurations. These simulators can be easily interfaced to the model generator CAPMOD.

## 6.6 Results

The accuracy of the numerical simulator in the model generator was verified by comparing the capacitance evaluated by the numerical simulation with Chang's analytical expression[46] available for a single line over a ground plane in a uniform dielectric. They agreed within 5%. More accuracy can be achieved by refining the mesh in the numerical simulator at the cost of higher CPU time.

Design-parameter-based analytical models for an example silicon technology having the following values of process parameters (refer to Fig. 6.11) were obtained from the model generator CAPMOD.

$$n = 3 \quad (6.12)$$

$$d_1 = 1.3\mu m, d_2 = 1.4\mu m, d_3 = 1.8\mu m, \quad (6.13)$$

$$\epsilon_{r1} = \epsilon_{r2} = \epsilon_{r3} = 3.9 \quad (6.14)$$

$$\epsilon_{rt} = 1.0 \quad (6.15)$$

$$t_1 = 0.6\mu m, t_2 = 0.8\mu m \quad (6.16)$$

As evident from the process parameters, there is an oxide-air interface above the second level of interconnect.  $t_1$  and  $t_2$  refer to the thicknesses of MET1 and MET2 in the example technology. The analytical models have been obtained for widths of lines in the range of 1 to 10  $\mu m$ , and separation between adjacent lines in the range of 1 to 5  $\mu m$ . (beyond this separation, the coupling capacitance falls below 10% of the capacitance to ground of each line which is the threshold used in this example). The capacitances for 2D configurations are in  $fF/\mu m$ , and for the 3D configuration (crossover) are in  $fF$ . The dimensions of design parameters are in  $\mu m$ .

The *design-parameter-based* analytical models presented in this section have been obtained from CAPMOD with a maximum error margin of about 10% with respect to data obtained from numerical simulations, although for some configurations the actual errors are much less. It is to be kept in mind that because of the inherent process variations associated with the interconnects, it may not make sense to obtain super-accurate models. However, the accuracy of the models can be controlled by controlling the error tolerance used inside the program.

For many of the configurations besides providing the analytical models, results of numerical simulation will be plotted to provide a better insight into the nature of variation of capacitances with the design parameters, which have already been explained in Section 6.4.

For the various parallel-line configurations it may so happen that when the coupling between two lines is weak, then the correction capacitance associated with each line is only a small fraction of capacitance to ground of the line if it was isolated. Thus, it is much more reasonable to estimate the error in the correction capacitance as a percentage of the capacitance to ground of the single line. This error criterion has been used for the correction capacitances.

In many of the models, a notation  $[f(x)]_{x < x_t}$  will be used to denote that the variation in  $f(x)$  is significant only for  $x < x_t \mu m$ . Hence  $f(x)$  can be treated as a constant equal to its value at  $x = x_t \mu m$  for  $x \geq x_t \mu m$ .

### 6.6.1 Models for Single Line

For the bottom layer, the models generated for bottom and top layers are given below. Bottom layer of metal:

$$C_0 = 0.081 + 0.027 w$$

Top layer of metal:

$$C_0 = 0.062 + 0.013 w$$

The fringe component which is reflected in the constants of the two expressions contributes a relatively larger fraction of capacitance for the top layer due to larger metal thickness of the top layer and the larger distance from the ground plane. These expressions should not be compared with any model which is only valid for uniform dielectric. For technology

under consideration the dielectric interface has a significant influence over self and coupling capacitances of the second metal.

### 6.6.2 Models for Crossover

The expressions obtained for the coupling and the correction capacitances are:

$$C_{12} = 0.16 + 0.093 w_1 + 0.071 w_2 + 0.048 w_1 w_2$$

$$C_{1c} = 0.126 + 0.064 w_2$$

$$C_{2c} = 0.162 + 0.067 w_1 + 0.018 w_1 w_2$$

The parallel-plate approximation underestimates the capacitance by a factor of about 7 for  $w_1 = w_2 = 1\mu m$ , and by a factor of about 1.2 for  $w_1 = w_2 = 10\mu m$ . Results of numerical simulation have been plotted in Fig. 6.14 and Fig. 6.15. For given widths of lines, the correction capacitance of bottom line ( $C_{1c}$ ) is found to be less than that of the top line ( $C_{2c}$ ) (as expected since the bottom line comes between the top line and the ground plane). Both correction capacitances are found to be less than the coupling capacitance.

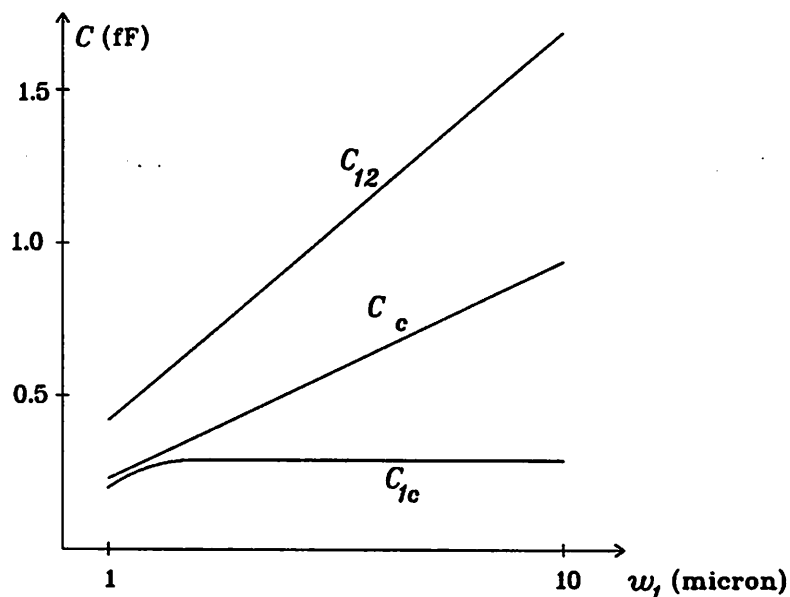


Figure 6.14: Variation of coupling and correction capacitances with width  $w_1$  in a crossover ( $w_2 = 1\mu m$ )

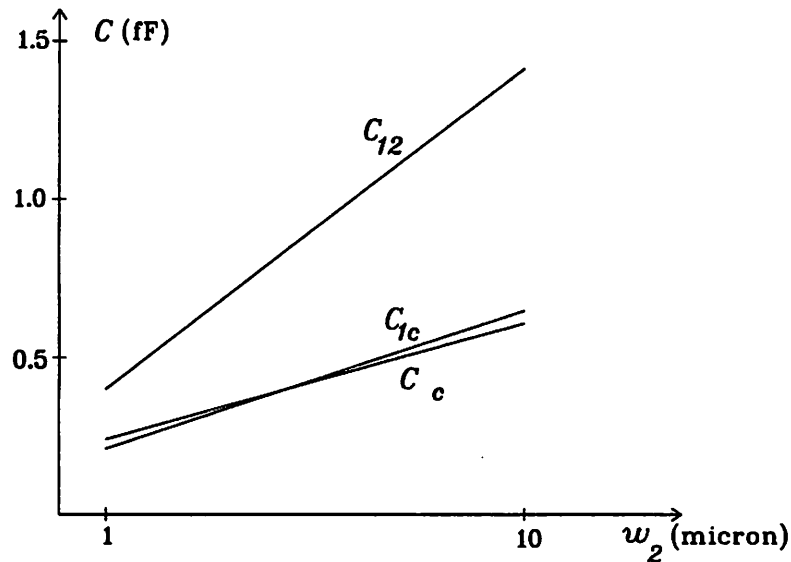


Figure 6.15: Variation of coupling and correction capacitances with width  $w_2$  in a crossover ( $w_1 = 1\mu m$ )

### 6.6.3 Models for Parallel Lines on the Same Layer

The models obtained for the per-unit-length coupling and correction capacitances of parallel lines on the bottom layer are:

$$C_{12} = -0.0048 + 0.106/s - 0.032/s^2 - [0.023/(s + w_1)]_{w_1 < 5} - [0.023/(s + w_2)]_{w_2 < 5}$$

$$C_{1c} = -0.011 + 0.187/(s + 2.7) - [0.052/(s + 2.7 + w_2)]_{w_2 < 5}$$

$$C_{2c} = -0.011 + 0.187/(s + 2.7) - [0.052/(s + 2.7 + w_1)]_{w_1 < 5}$$

A second degree polynomial in  $(1/s)$  has been obtained for the model of coupling capacitance, but only a first degree polynomial in  $(1/(s + 2.7))$  was sufficient for the correction capacitances, since the errors in the correction capacitances have relatively less effect on the overall error in the total capacitance to ground. The models for parallel lines on the top layer are similar.

### 6.6.4 Models for Parallel Lines on Different Layers

For the nonoverlapping configuration (a), the following expressions were obtained.

$$C_{12} = -0.01 + 0.18/(s + 0.9) - 0.074/(s + 0.9)^2 - [0.037/(s + 0.9 + w_1)]_{w_1 < 6}$$

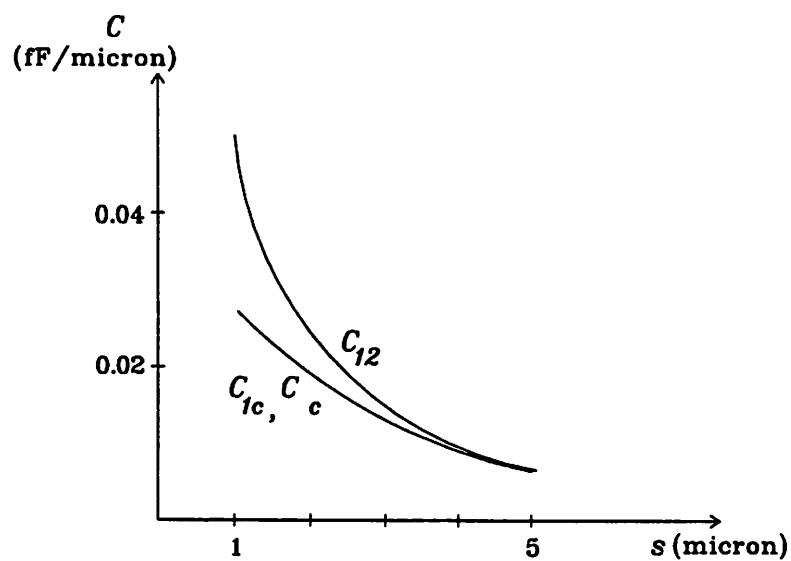


Figure 6.16: Variation of coupling and correction capacitances with the separation  $s$  for parallel lines on bottom layer ( $w_1 = w_2 = 1\mu m$ )

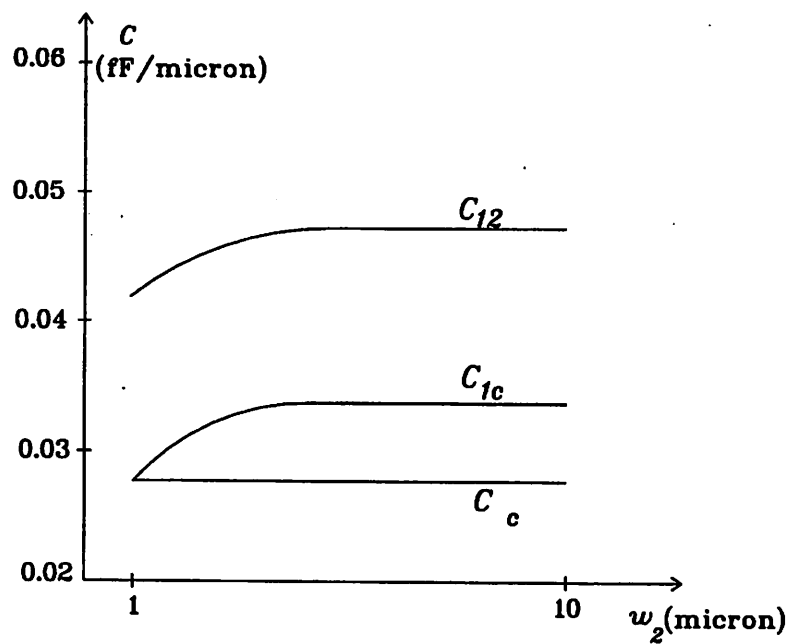


Figure 6.17: Variation of coupling and correction capacitances with the width  $w_2$  for parallel lines on bottom layer ( $s = w_1 = 1\mu m$ )

$$C_{1c} = 0.007 + 0.262/(s + 5.5) - [0.08/(s + 5.5 + w_2)]_{w_2 < 3}$$

$$C_{2c} = -0.023 + 0.327/(s + 2.3) - 0.317/(s + 2.3)^2 - [0.079/(s + 2.3 + w_1)]_{w_1 < 6}$$

Besides the absence of the singularity at  $s = 0$  for the coupling capacitance, a difference between these models and those for parallel lines on the same layer is that these models are not symmetric with respect to the two lines (i.e. they will not remain the same when subscripts 1 and 2 are interchanged), due to the inherent asymmetry of the configuration.

For overlapping configuration (b), the following expressions were obtained.

$$C_{12} = 0.094 + 0.046p - [0.041/(e_l + 1.2)]_{e_l < 6} - [0.040/(e_r + 1.2)]_{e_r < 3}$$

$$C_{1c} = 0.072 - [0.043/(p + 1.4)]_{p < 3} + [0.038/(e_l + 1.8)]_{e_l < 6} - [0.096/(e_r + 1.8)]_{e_r < 3}$$

$$C_{2c} = 0.051 + 0.013p - [0.088/(e_l + 1.6)]_{e_l < 6} + [0.032/(e_r + 1.6)]_{e_r < 3}$$

Results of numerical simulation showing the variation of coupling and correction capacitances with  $p$  and  $e_l$  have been plotted in Fig. 6.18 and Fig. 6.19. The variations with  $e_r$  are similar to that with  $e_l$ , the only difference being that for small  $e_r$ ,  $C_{2c}$  decreases, and  $C_{1c}$  increases with  $e_r$ .

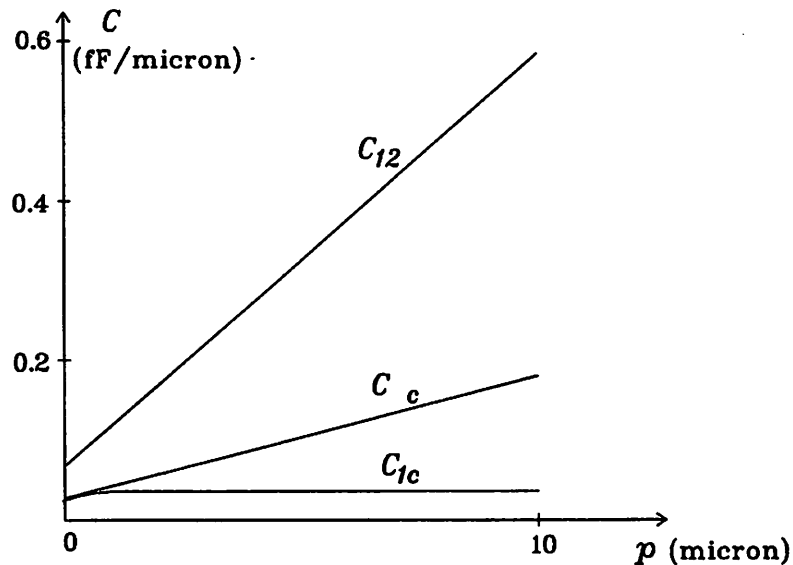


Figure 6.18: Variation of coupling and correction capacitances with  $p$  for parallel lines in configuration (b) ( $e_l = e_r = 1\mu m$ )

The models for configurations (c) and (d) are similar, but with only different values of coefficients. It took a substantial amount of CPU time to generate the analytical models



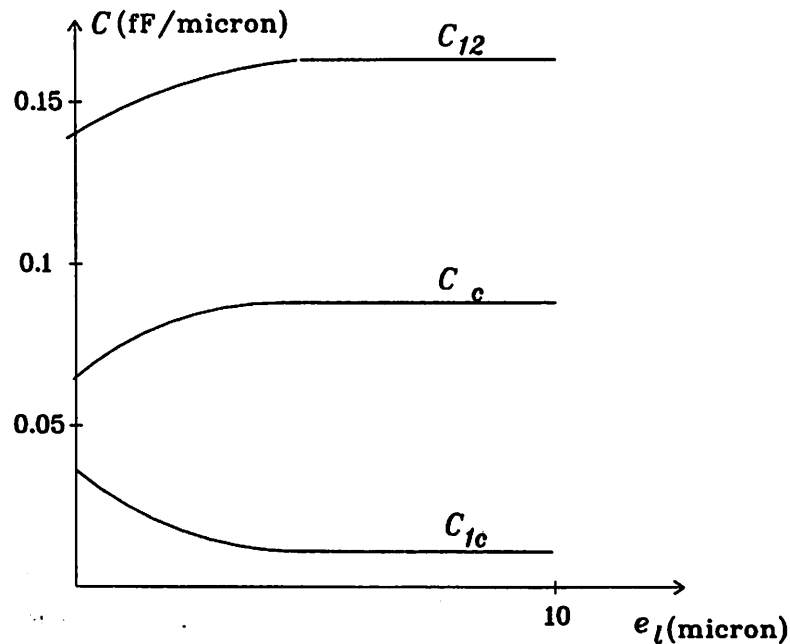


Figure 6.19: Variation of coupling and correction capacitances with  $e_l$  for parallel lines in configuration (b) ( $p = 2\mu m$ ,  $e_r = 0\mu m$ )

(about 40,000 sec on VAX8650). However, it has to be done only once for each technology.

## 6.7 Summary

In this chapter, a tool for automatically generating analytical models of interconnect capacitances was presented. It uses a partial knowledge of the flux components associated with a configuration to choose a suitable form of analytical expression, and then uses curve-fitting techniques to obtain analytical models. For each coupled configuration, the form for the coupling capacitance is chosen based on a decomposition of the mutual flux associated with the two lines. The form for the correction capacitance of each line is decided based on a decomposition of its flux intercepted by the other line. A design-parameter-based modeling is pursued, since often it is desired to perform large number of evaluations of a capacitance in a layout, with a fixed set of process parameters, but for varying values of

design parameters. Generation of analytical models for a process with two levels of metals was illustrated.

## Chapter 7

# Conclusions and Future Directions

### 7.1 Conclusions

The traditional approach towards layout design of analog circuits involves expensive layout-extraction-simulation iterations and is often very inefficient. This thesis proposed a novel performance-constrained approach towards automatic layout design, in which constraints are derived automatically from performance constraints and then used to drive the layout tools. Hence, in this approach the need for expensive layout iterations is reduced. All the previous approaches towards automatic layout design of analog and mixed analog/digital circuits did not cope directly with the performance constraints associated with the circuits.

A novel flexibility-based algorithm for generating bounding constraints on parasitics from the specified performance constraints, was presented in Chapter 4. The problem of distributing the parasitic constraints for maximizing the flexibility of the layout tools is formulated as a quadratic programming problem, with quadratic cost function and linear performance constraints. Linear approximations using sensitivities are used to model the performance constraints, since the goal is to keep performance degradation small around the nominal values. For the test examples illustrated in this chapter, it was noted that these approximations are acceptable. Moreover, since standard circuit simulators have sensitivity computation capabilities, this approach is also efficient from an implementation point of view. Matching constraints on parasitics are derived from the matched-mode-pair information of the circuit. Worst-case expressions for sensitivities were derived taking process variations and mismatch (between matched parasitics) into account. These expressions were

used to model the performance constraints, and the importance of this worst-case modeling was illustrated for matched parasitics. An elimination procedure was presented for selecting the critical parasitics based on sensitivities and performance constraints. It was noted that a large fraction of parasitics can be eliminated at this stage. As a result, bounding constraints can be imposed on a relatively small set of critical parasitics without unnecessarily overloading the constraint-driven layout tools.

Work on constraint-driven layout design was presented in Chapter 5. Constraint-driven channel routing was described in detail and an overview of constraint-driven area routing and placement was provided. Algorithms were presented for mapping the constraints on a set of critical coupling capacitances into constraints in the Vertical-Constraint(VC) graph of a channel. These algorithms involve directing undirected edges, adding directed edges and increasing the weights of edges in the VC graph, in order to meet crossover constraints between orthogonal segments and adjacency constraints between parallel segments, while attempting to cause minimum increase in the channel height. Use is made of shield nets when necessary. A technique to achieve almost-perfect mirror symmetry in the channel was presented for pairs of nets with overlapping spans. For the test examples a certain increase in channel area was observed due to the imposition of the parasitic constraints. However, since area of an analog circuit is usually dominated by the devices, the increase in routing area has relatively less impact on total area. Moreover, in analog circuits area can be traded off when it ensures satisfaction of performance constraints of the circuit. It was also observed that performance constraints are met even if some of the parasitic constraints are not met. This is because other parasitics affecting the performance functions may meet their respective constraints by large margins.

Fast evaluation of interconnect parasitics is essential for constraint-driven layout design and also for final extraction after layout design. Unfortunately parallel-plate approximation causes gross error in estimation of line-to-line and line-to-line coupling capacitances. The analytical models suggested previously for capacitances are usually for single lines and moreover make assumptions about the process. In Chapter 6, a tool for generating analytical models of interconnect capacitances automatically was presented. It uses a partial knowledge of the flux components associated with a configuration to choose a suitable form of analytical expression, and then uses curve-fitting techniques to obtain the analytical models. The configurations which can be handled currently by this framework are (a) single-line (b) crossing lines (c) parallel lines on the same layer and (d) parallel lines in different layers

(both overlapping and nonoverlapping). For each coupled configuration, the form for the coupling capacitance is chosen based on a decomposition of the mutual flux associated with the two lines. The form for the correction capacitance of each line is decided based on a decomposition of its flux intercepted by the other line. A design-parameter-based modeling is pursued, since often it is desired to perform a large number of evaluations of parasitic capacitances in a layout, with a fixed set of process parameters, but for varying values of design parameters. Although it takes a significant amount of CPU time to generate the analytical models, it has to be done only once for each technology.

Many of the CAD tools mentioned can be useful in *aiding* designers, even when the layout is designed manually. For example, the parasitic constraint generator can be used by designers to identify critical parasitics, and obtain their associated sensitivities as well as the bounds which the parasitics have to satisfy in manual layout design. The analytical-model generator for interconnect capacitances can be used by the designers to obtain expressions which can guide them in the layout design.

## 7.2 Future Directions

Currently, the constraints which can be generated for automatic layout design (by the constraint generator PARCAR) are those associated with analog circuits in an analog or a mixed A/D system. In recent years there has been active research in performance-driven placement and routing for digital circuits considering the delays associated with the interconnects. However, it would be worthwhile to consider also the crosstalk between interconnects during layout design of digital circuits. For a technology with very small value of minimum feature size, crosstalk can even change the logic state of a net due to the switching of an adjacent net. Hence developing constraint-generation capability in PARCAR for coupling capacitances in digital circuits will be a useful effort. The constraint-driven routers already developed can then be used for routing.

Another useful contribution in automated layout design will be considering of high-frequency phenomena such as the transmission-line effect for high-speed digital and microwave integrated circuits. Many high-frequency phenomena are of concern even at moderately high speeds of operation for PC Boards and Multichip Modules, due to the larger physical dimensions encountered. There has been considerable amount of research in recent years for efficient simulation of these high-frequency phenomena. However, dealing

with them in automated layout design remains a wide-open research area to be explored.

Considerable amount of work can be continued on interconnect modeling. Developing an inductance-model generator similar to the analytical-model generator CAPMOD should be the next step in this direction. The capacitance and inductance models can be used in constraint-driven layout design and for fast and accurate parasitic extraction.

Another interesting problems is developing sensitivity analysis algorithms for high-level and mixed-mode simulations, and using them for performance-constrained layout design. Currently, the parasitic-constraint generator PARCAR is interfaced to SPICE3 and SWAP (a switch-capacitor simulator) with sensitivity-analysis capabilities). It would be a worthwhile effort to interface high-level and mixed-mode simulators with sensitivity analysis capabilities to the constraint generator PARCAR, thus extending the applicability of the performance-constrained approach to mixed analog/digital circuits.

# Bibliography

- [1] R.J. Bowman and D.J. Lane "A knowledge-based System for Analog Integrated Circuit Design," *Proc. IEEE International Conference on Computer-Aided Design*, pp. 210-212, 1985
- [2] W.J. Helms and K.C. Russel et al., "A Switched Capacitor Filter Compiler" *Proc. IEEE Custom Integrated Circuits Conference*, 1986, pp. 125-128.
- [3] F.M. El-Turkey and R.A. Nordin "BLADES: An Expert System for Analog Circuit Design," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 552-555, 1986
- [4] R. Harjani, R.A. Rutenbar and L.R. Carley, "A Prototype Framework for Knowledge-Based Analog Circuit Synthesis," *Proc. 24th ACM/IEEE Design Automation Conference*, pp. 42-49, 1987
- [5] H.Y. Koh, C.H. Sequin, and P.R. Gray, "Automatic Synthesis of Operational Amplifiers Based on Analytic Circuit Models" *Proc. IEEE ICCAD*, 1987, pp. 548-551.
- [6] M.G. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B.L.A.G. Goffart, E.A. Vittoz, S. Cserveny, C. Meixenberger, G.V.D. Stappen and H. J. Oguey, "IDAC: An Interactive Design Tool for Analog CMOS Circuits" *Proc. IEEE Journal of Solid State Circuits* Vol. SC-22, No. 6, Dec. 1987, pp. 1106-1116.
- [7] E. Berkcan, M. d'Abreu and W. Laughton et al., "Analog compilation based on successive decompositions", *Proc. ACM/IEEE Design Automation Conference* 1988, pp. 369-376.
- [8] G.D. Hachtel, M.R. Lightner and H.J. Kelly, "Application of the Optimization program AOP to the Design of Memory Circuits", *IEEE Transactions on Circuits and Systems*, June 1975, pp. 496-503.
- [9] ASTAP Advanced Statistical Analysis Program. *IBM Program Product Document* SH20-1118-0 edition, 1973.

- [10] G.D. Hachtel and P. Zug "Circuit Design and Optimization System," - *User's guide Technical Report, IBM Yorktown Research Facility*, 1981
- [11] W.T. Nye, D. Riley, A. Sangiovanni-Vincentelli and A.L. Tits, "DELIGHT.SPICE: An optimization-Based System for the Design of Integrated Circuits and Systems," *IEEE Trans. on CAD*, Vol. 7, No. 4, Apr. 1988, pp. 501-519.
- [12] J. Shyu and A. Sangiovanni-Vincentelli "ECSTASY: A New Environment for IC Design Optimization" *Proc. IEEE International Conference on Computer-Aided Design*, 1988, pp. 484-487.
- [13] C.D. Kimble, A.E. Dunlop, G.F. Gross, V.L. Hein M.Y. Luong, K.J. Stern and E.J. Swanson, "Autorouted Analog VLSI," *Proc. IEEE Custom Integrated Circuits Conference*, 1985, pp. 72-78.
- [14] P.E. Allen, E.R. Macaluso, S.F. Bily and A. Nedungadi, "AID2: An Automated Analog IC Design System" *Proc. IEEE Custom Integrated Circuits Conference*, 1985, pp. 498-501.
- [15] P.E. Allen and P.R. Barton "A Silicon Compiler for Successive A/D and D/A Converters" *Proc. IEEE Custom Integrated Circuits Conference*, 1986, pp. 552-555.
- [16] H. Yaghtiel, A. Sangiovanni-Vincentelli, and P.R. Gray, "A Methodology for Automated Layout of Switched-Capacitor Filters," *Proc. IEEE International Conference on Computer-Aided Design*, pp. 444-447, 1986
- [17] J. Tronteli, L. Tronteli, T. Pleterssek, G. Shenton, M. Robinson, K. Floyd, and C. Jungo *et al.*, "Expert System for automated mixed analog/digital layout compilation" *Proc. IEEE Custom Integrated Circuits Conference*, 1987, pp. 165-167.
- [18] J. Rijmenants, J.B. Litsios, T.R. Schwarz and M.G.R. Degrauwe "ILAC: An Automated Layout Tool for Analog CMOS Circuits" *IEEE Jor IEEE Journal of Solid State Circuits* Vol. 24, no.2, pp.436-442, April 1989.
- [19] H.Y. Koh, C.H. Sequin, and P.R. Gray, "Automatic Layout Generation for CMOS Operational Amplifiers" *Proc. IEEE ICCAD*, 1988, pp. 548-551.
- [20] D. Garrod, R.A. Rutenbar and L.R. Carley, "Automatic Layout of Custom Integrated Circuits in ANAGRAM," *Proc. IEEE ICCAD*, Nov. 1988, pp. 544-547.
- [21] M. Kayal, S. Piguët, M. Declercq, and B. Hochet *et al.*, "SALIM: A Layout Generation Tool for Analog ICs" *Proc. IEEE Custom Integrated Circuits Conference*, 1988, pp. 7.5.1-7.5.4.



- [22] D.J. Chen, J.C. Lee and B.J. Sheu, "SLAM: A Smart Analog Module Generator for Mixed Analog-Digital VLSI Design" *Proc. IEEE International Conference on Computer Design* 1989, pp. 24-27
- [23] J.L. Burns and A.R. Newton, "SPARCS: A new Constraint-Based IC Symbolic Layout Spacer," *Proc. IEEE Custom Integrated Circuits Conference*, May 1986, pp. 534-539
- [24] L. Rijnders, P. Six and H.J. DeMan "Design of a Process-Tolerant Cell Library for Regular Structures Using Symbolic Layout and Hierarchical Compaction," *IEEE J. of Solid-State Circuits*, Vol. 23, No. 3, 1988, pp. 714-721.
- [25] R. Okuda, T. Sato, H. Onodera and K. Tamaru, "An Efficient Algorithm for Layout Compaction Problem with Symmetry Constraints," *Proc. ICCAD*, Nov. 1989, pp. 148-151.
- [26] M. Kawakita and T. Watanabe "Analog Layout Compaction with Clean-up Function" *Proc. Transactions of IEICE*, Vol. E71 No.12, pp.1243-1252, Dec 1988.
- [27] M. Mogaki, N. Kato, Y. Chikami, N. Yamada and Y. Kobayashi "An Automatic Layout System for Analog LSI's" *Proc. IEEE ICCAD*, Nov. 1989, pp. 450-453.
- [28] R.S. Gyurcsik and J.-C. Jeen, "A Generalized Approach to Routing Mixed Analog and Digital Signal Nets in a Channel," *IEEE Journal of Solid-State Circuits*, Vol 24, No. 2, Apr. 1989, pp. 436-442.
- [29] E. Malavasi, M. Chilanti and R. Guerrieri, "A General Router for Analog Layout" *Proc. of Compeuro*, 1989, pp. 549-551.
- [30] S.K. Hong and P.E. Allen, "Performance-Driven Analog Layout Compiler" *Proc. IEEE International Symposium on Circuits and Systems*, May 1990, pp. 835-838.
- [31] U. Choudhury and A. Sangiovanni-Vincentelli, "Use of Performance Sensitivities in Routing of Analog Circuits," *Proc. International Symposium on Circuits and Systems*, May 1990, pp.348-351.
- [32] M. Itoh and H. Mori, "A Layout Generating and Editing System for Analog LSIs," *Proc. IEEE International Symposium on Circuits and Systems*, 1990, pp. 843-846.
- [33] S. Piguet, F. Rahali, M. Kayal, E. Zysman and M. Declercq, "A New Routing Method for Full Custom Analog IC's" *Proc. Custom Integrated Circuits Conference* 1990, pp. 27.7.1-27.7.4
- [34] Y. Shiraishi, M. Kimura, K. Kobayashi, T. Hino and M. Seriuchi and M. Kusaoke "A High-Packing Density Module Generator for Bipolar Analog LSIs" *Proc. IEEE ICCAD*, 1990, pp. 194-197.

- [35] I. Harada, H. Kitazawa and T. Kaneko, "A Routing System for Mixed A/D Standard Cell LSI's" *Proc. IEEE ICCAD*, 1990, pp. 378-381.
- [36] G. Jusuf, P.R. Gray and A. Sangiovanni-Vincentelli, "CADICS-Cyclic Analog-to-Digital Converter Synthesis" *Proc. IEEE ICCAD*, 1990, pp. 286-289.
- [37] J.M. Cohn, D.J. Garrod, R.A. Rutenbar and L.R. Carley, "KOAN/ANAGRAMII: New Tools for Device-Level Analog Placement and Routing", *IEEE Journal of Solid State Circuits*, March 1991, vol. 26, pp. 330-342
- [38] U. Choudhury, "Sensitivity Computation in SPICE3," *Masters Thesis*, U.C. Berkeley, Dec 1988.
- [39] T.H. Nguyen and H. Guo "Performance Sensitivity Computation for Interconnect Parasitics" *EECS 244 project report*, U.C. Berkeley, Dec 1990.
- [40] D.E. Hocevar and P. Yang, "Practical Issues for Implementing Transient Sensitivity Computation," *Proc. IEEE ICCAD*, 1984, pp. 1-3.
- [41] "The Switched Capacitor Network Simulator SWAP Reference Manual," *Silvar-Lisco*, Release 2.0, Nov. 1983.
- [42] S.W. Director and R.A. Rohrer, "The Generalized Adjoint Network Sensitivities," *IEEE Trans. Circuit Theory*, Vol CT-16, Aug 1969, pp. 318-323.
- [43] C.M. Sakkas "Potential Distribution and Multi-Terminal DC Resistance Computation for LSI Technology", *IBM Journal of Research and Development*, Vol. 23, No. 6, November 1979.
- [44] M.V. Schneider, "Microstrip lines for Microwave integrated circuits" *Bell Syst. Tech. J.*, vol. 48, no 5, May 1969, p. 1421.
- [45] H.A. Wheeler, "Transmission line Properties of a sheet on a plane" *IEEE Trans. Microwave Theory Tech.*, vol. MTT-25, no, 8 Aug. 1977 p. 631.
- [46] W.H. Chang, "Analytical IC Metal-Line Capacitance Formulas" *IEEE Trans. on Microwave Theory and Techniques*, Vol. MTT-24, Sep. 1976, pp. 608-611, also Vol. MTT-25, pp.712 August 1977
- [47] M.I. Elmasry, "Capacitance Calculation in MOSFET VLSI" *IEEE Electron Device Lett.*, vol.EDL-3, 1981, pp.6-7.
- [48] 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, 1982, pp.391-393.

- [49] T. Sakurai and K. Tamaru, "Simple formulas for two- and three-dimensional capacitances," *IEEE Trans. Electron Devices*, vol. ED-30, 1983, pp.183-185.
- [50] N. Meijs and J.T. Fokkema, "VLSI circuit reconstruction from mask topology," *Integration*, vol. 2,no. 2, 1984, pp.85-119.
- [51] K.C. Gupta, R. Garg and R. Chadha, "Computer Aided Design of Microwave Circuits," *Norwood, MA: Artech House*. 1981 pp.76-79.
- [52] J. Chern, Texas Instruments, Dallas, *Private Communication*.
- [53] B. Donecker, Hewlett-Packard, Santa Rosa, *Private Communication*.
- [54] T.J. Rivlin, "An Introduction to the Approximation of Functions", *Blaisdell Publishing Company*, 1969, pp. 42. *Norwood, MA: Artech House*. 1981 pp.76-79.
- [55] E. Barke, "Line-to-Ground Capacitance Calculation for VLSI: A Comparison," *IEEE Transactions on Computer-Aided Design* 1976, Vol. 7, No. 2, Feb. 1988, pp. 295-298.
- [56] A. E. Ruehli and P.A. Brennan, "Efficient Capacitance Calculations for Three Dimensional Multiconductor Systems," *IEEE Trans. Microwave Theory Tech.* MTT-21, 1973, pp. 76-82.
- [57] P.E. Cottrell and E.M. Buturla, "VLSI Wiring Capacitance," *Proc. IBM Journal of Research and Development*, 1985, pp. 277-287.
- [58] R.H. Uebbing and M. Fukumma, "Process-Based Three-Dimensional Capacitance Simulation-TRICEPS" *IEEE Transactions on CAD*, Vol. CAD-5, No. 1, Jan. 1986, pp. 215-998.
- [59] A. Seidel *et.al.* , "CAPCAL-A 3-D Capacitance Solver for Support of CAD systems," *IEEE Trans. on CAD*, Vol. 7, No. 5, May. 1988, pp. 556.
- [60] Z. Ning and P. M. Dewilde , "SPIDER: Capacitance Modelling for VLSI Interconnections," *IEEE Trans. on CAD*, Vol. 7, No. 12, Dec. 1988, pp. 1221-1228.
- [61] P. M. Dewilde and Z. Ning , "Models for Large Integrated Circuits," *Kluwer Academic Publishers* 1990.
- [62] R. Guerrieri and A. Sangiovanni-Vincentelli, "Three-Dimensional Capacitance Evaluation on a Connection Machine" *IEEE Trans. on CAD*, Vol. 7, 1988, pp. 1125-1133.
- [63] K. Nabors and J. White , "A Fast Multipole Algorithm for Capacitance Extraction of Complex 3-D Geometries," *Proc. Custom Integrated Circuits Conference* 1989
- [64] C. Wei, R. F. Harrington, J.R. Mautz and T.K.Sarkar, "Multiconductor transmission lines in multilayered dielectric media," *IEEE Trans. Microwave Theory Tech.* vol. MTT-32, Apr. 1984, pp. 439-450.

- [65] A. E. Ruehli, "Inductance Calculation in a Complex Integrated Circuit Environment". *IBM Journal of Research and Development*, 1972, pp. 470-481.
- [66] A. E. Ruehli, "Survey of Computer-Aided Electrical Analysis of Integrated Circuit Interconnections". *IBM Journal of Research and Development*, Vol. 23, No. 6, Nov. 1979, pp. 626-639.
- [67] W.T. Weeks, L.L. Wu, M.F. McAllister and A.Singh, "Resistive and Inductive Skin Effect in Rectangular Conductors," *IBM Journal of Research and Development*, Vol. 23, No. 6, Nov. 1979, pp. 626-639.
- [68] A. Sangiovanni-Vincentelli, "Automatic Layout of Integrated Circuits," *Design Systems for VLSI circuits, Logic Synthesis and Silicon Compilation*, Martinus Nijhoff Publishers, 1987, pp. 113-195.
- [69] T. Ohtuski(editor), "Layout Design and Verification," North Holland, 1986
- [70] L. Stockmeyer, "Optimal Orientations of Cells in Slicing Floor-Plan Design", *Information and Control* vol. 59, pp. 91-101, 1983.
- [71] R. Otten, "Layout Compilation," *Design Systems for VLSI ckts, Logic Synthesis and Silicon Compilation*, Martinus Nijhoff Publishers, 1987, pp. 439-472.
- [72] M. A. Breuer, "Min-Cut Placement," *J. Des. Automat. Fault Tolerant Comput.*, pp. 343-362, Oct. 1977.
- [73] U. Lauther, "A Min-Cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation", *Journal of Digital Systems* Vol. IV, Issue 1, pp. 21-34, 1980.
- [74] B. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", *Bell Systems Technical Journal*. vol. 49, no. 2, pp. 291-307, Feb. 1970.
- [75] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, N.4598, pp. 671-680, May 1983.
- [76] C. Sechen and A. Sangiovanni-Vincentelli "Timber Wolf: A placement System for Integrated Circuits", *Journal of Solid-State Circuits*, May 1985
- [77] A. Hashimoto and J. Stevens, "Wire Routing by Optimizing Channel Assignment within Large Apertures," *Proc. 8th Design Automation Workshop*, 1971, pp. 155-169.
- [78] W.M. Dai, T. Asano, and E.S. Kuh, "Routing Region Definition and Ordering Scheme for Building Block Layout", *IEEE Trans. Computer-Aided Design* vol. CAD-4, pp. 189-197, July 1985.
- [79] T. Yoshimura and E.S. Kuh, "Efficient Algorithms for Channel Routing", *IEEE Trans. on Computer-Aided Design*, pp. 25-36, Jan 1982

- [80] R. Rivest and C. Fiduccia, "A Greedy Channel Router", *Proc. 19th Design Automation Conference*, pp. 418-424, 1982.
- [81] M. Burstein and R. Pelavin, "Hierarchical Wire Routing", *IEEE Trans. on Computer-Aided Design* pp. 223-234, Oct. 1983.
- [82] J. Reed, A. Sangiovanni-Vincentelli and M. Santomauro "A New Symbolic Channel Router: YACR2," *IEEE Trans. on CAD* Vol. CAD-4, No.3, July 1985, pp. 208-219.
- [83] C. Y. Lee "An algorithm for path connections and its Application", *IRE Trans. Electron. Comp.* pp. 346-365, Sep. 1961.
- [84] H. Shin and A. Sangiovanni-Vincentelli, "MIGHTY: A Rip-up and Re-route Detailed Router", *Proc. of International Conference on Computer-Aided Design*, Nov. 1986, pp. 2-5.
- [85] D.W. Hightower, "A solution to Line-Routing Problem on the Continuous Plane" *Proc. 6th Design Automation Workshop*, 1969, pp. 1-24.
- [86] D. Braun, J. Burns, S. Devadas, H.K. Ma, K. Mayaram, F. Romeo and A. Sangiovanni-Vincentelli "Chameleon: A New Multi-Layer Channel Router", *Proc. of 23rd Design Automation Conference*, 1986, pp. 495-502.
- [87] Y. E. Cho, A.J. Korenjak, and D.E. Stockton, "FLOSS: An approach to Automated Layout for High-Volume Designs", *Proc. of 14th Design Automation Conference*, June 1977, pp. 138-141.
- [88] R. P. Larsen, "Computer-Aided Preliminary Layout Design of Customized MOS Arrays," *IEEE Transactions on Computers*, Vol. C-20, No. 5, May 1971, pp. 512-523.
- [89] J. D. Williams, "STICKS: A graphical compiler for high level LSI design", *AFIPS Conference Proceedings*, vol. 47, June 1978, pp. 289-295.
- [90] M. Y. Hsueh and D.O. Pederson "Computer-Aided Layout of LSI circuit Building-Blocks", *Proc. of IEEE International Symposium on Circuits and Systems*, June 1979, pp.474-477.
- [91] M. Schlag, Y.Z. Liao and C.K. Wong "An Algorithm for Optimal Two-Dimensional Compaction of VLSI layouts," *Integration VLSI Journal 1*, 1983, pp. 197-209.
- [92] H. Shin, A. Sangiovanni-Vincentelli and C. Sequin, "Two-Dimensional Compaction by Zone Refining", *Proc. 23rd ACM/IEEE Design Automation Conference*. pp. 115-122, June 1986

- [93] E. Malavasi, U. Choudhury and A. Sangiovanni-Vincentelli, "A Routing Methodology for Analog Integrated Circuits," *Proc. IEEE International Conference on Computer-Aided Design*, Nov. 1990, pp. 202-205.
- [94] D. Deutsch, "A Dogleg Channel Router," *Proc. 13th Design Automation Conference*, 1976, pp. 425-433.
- [95] H.H. Chen and E.S. Kuh, "Glitter: A Gridless Variable-Width Channel Router," *IEEE Transactions on CAD*, Vol. CAD-5, No. 4, Oct. 1986, pp. 459-465.
- [96] E. Malavasi, "A User Interface for Performance Constraint Specification" *EECS 219 Project Report*, U.C. Berkeley, Dec 1991.
- [97] A.R. Alvarez, B.L. Abdi, D.L. Young, H.D. Weed, J. Teplik, and E.R. Herald, "Application of statistical design and response surface methods to computer-aided VLSI device design," *IEEE Trans. Computer-Aided Design*, vol. CAD-7, pp. 272-288, Feb. 1988.
- [98] K.K. Low and S.W. Director, "An efficient methodology for building macromodels of IC fabrication processes," *IEEE Trans. Computer-Aided Design*, vol. CAD-8, pp. 1299-1313, Dec. 1989.
- [99] T.K. Yu, S.M. Kang, I.N. Hajj, and T.N. Trick, "Statistical performance modelling and parametric yield estimation of MOS VLSI," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 1013-1022, Nov. 1987.
- [100] T.K. Yu, S.M. Kang, J. Sacks, and W.J. Welch, "Parametric yield optimization of MOS integrated circuits by statistical by statistical modelling circuit performances," *Technical Report No. 27*, Dept. of Statistics, University of Illinois, Champaign, July 1989.
- [101] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn, "Design and analysis of computer experiments," *Statistical Science* Nov. 1989.
- [102] L. Milor, "Fault-driven Analog Testing" *Ph.D. dissertation, Chapter 3*, Dept. of Electrical Engineering, U.C. Berkeley, May. 1992.
- [103] A. Casotto, F. Romeo and A. Sangiovanni-Vincentelli, "A Parallel Simulated Annealing Algorithm for the Placement of Macro-Cells," *IEEE Trans. on CAD* Vol. CAD-6, No. 5, Sep. 1987, pp. 838-847.
- [104] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto and A. Sangiovanni-Vincentelli, "A Constraint-Driven Placement Methodology for Analog Integrated Circuits," *accepted in IEEE Custom Integrated Circuits Conference*, 1992.

- [105] M. Marek-Sadowska and T.T. Tarng, "Single-Layer Routing for VLSI: Analysis and Algorithms," *IEEE Trans. on Computer-Aided Design*, vol. CAD-2, no. 4, pp. 246-259, Oct. 1983.

## Appendix A

# User's Manual of PARCAR

### NAME

PARCAR - a parasitic constraint generator for analog circuits.

### SYNTAX

```
parcar [-m maxmin_file] [-s sens_file] [-p perf_file] [-c layout_file] [-l cap_library_file]
[-o out_file]
```

### DESCRIPTION

PARCAR is a parasitic constraint generator for automatic layout design of analog circuits. The program has two modes of operation (a) a normal mode and (b) a special mode. By default the program operates in the normal mode. If “-c” option is used, it operates in the special mode (also called the channel-routing mode). The special mode has been developed for interfacing PARCAR to a constraint-driven channel router ART. In both the modes the program requires a performance file which contains the performance constraints of the circuit. By default the name of this file is “perf\_file” and it can be changed using the “-p” option. If a “-s” is not used the program computes the sensitivities of performance functions calling the SPICE3 simulator before generating the constraints. If a “-s” option is used then the program reads the sensitivity file (by default is “sens\_file” but can be another file whose name is used after “-s”). The sensitivity file contains the sensitivities of all the performance functions with respect to the parasitics. The “-s” option is used to give flexibility of obtaining the sensitivities using some other simulator and then running PARCAR to obtain the constraints.



In the normal mode the program requires a "maxmin\_file" consisting of estimates of maximum and minimum values of parasitics. In this mode, the program needs the "layout\_file" which has dimensions of the unrouted channels. In the special mode the program also requires a capacitance library file (default name "cap\_library\_file"). This file contains the coefficients used for calculating the coupling capacitances and can be automatically generated by the model generator CAPMOD. By default the program's output is standard output. It can be directed to any output file using "-o" option.

Expected syntax for the various files of PARCAR is now discussed. In this description integer refers to the value in integer format, float refers to the value in floating point format and string refers to a name in string format. The perf\_file should have the following syntax.

"num\_performance (integer), simulator (string)" - number of performance functions and the SPICE3 simulator name (full path);

"number of nodes (integer) out\_node1 (integer) out\_node2 (integer)" number of nodes in the circuit, name of first output node, name of second output node in the circuit. This should be followed by correct number of lines giving the names of nodes in the circuit.

"number of matched pairs (integer)" - number of matched pairs of nodes in the circuit followed by correct number of lines providing the names of nodes to be matched

"maximum process variation in percent (float)" - maximum process variation in a routing parasitic expressed as percent.

"maximum mismatch in percent (float)" - maximum mismatch between (nominally) matched parasitics in percent.

"performance type num\_constraints and sim\_file:(string) (integer) (integer) (string)" - name of the performance function, the type is always 0 (this option is there to classify different types of performance functions in future implementations), num\_constraints can be 1 or 2 depending on whether the performance has constraint in only one direction or both the directions, sim\_file is the name of the spice input deck to be used for simulating the performance. Different performance functions can have different spice input decks. This line should be followed by correct number of line(s) describing the constraint(s) in the following format.

"maximum positive(negative) change (float)": the maximum change allowed in the positive(negative)direction from the nominal value of the performance function.

"performance end (integer)"- to denote the end of performance description followed by an

integer equal to number of performance functions.

The syntax for the sensitivity file is as follows:

“number of nonzero (string) sensitivities = (integer)”

where (string) is the name of the performance function and (integer) is the number of parasitic capacitances whose sensitivities are provided. This line should be followed by the correct number of lines giving the pairs of nodes between which the capacitance is considered and the associated sensitivity. This format should be repeated for each performance of interest.

The syntax for the maxmin file is:

“number of nets (integer)”: number of nets in the layout (should be equal to the number of nodes in the circuit)

“default max\_self\_cap min\_self\_cap max\_coup\_cap min\_coup\_cap (float) (float) (float) (float)”: gives the default values of maximum self capacitance, minimum self capacitance, maximum coupling capacitance and minimum coupling capacitance respectively.

“number of net pairs (integer)”: the number of net pairs for which conservative estimates of maximum and minimum capacitances are provided. This line should be followed by the correct number of lines, each line providing net names of a pair, the maximum and the minimum capacitance estimate.

## COMPILATION

For compilation, type “make” in first in each subdirectory of Constraint except Constraint/main. Then type “make” in Constraint/main.

## EXAMPLES

See “choudhur/parcar/filter” and “choudhur/parcar/opamp” directories for examples.

## Appendix B

# User's Manual of ART

### NAME

ART - a channel router for analog and mixed analog/digital circuits.

### SYNTAX

art [option] [input\_file]

### DESCRIPTION

ART is a gridless channel router which can be driven by bounding constraints on coupling capacitances and matching constraints for matched pairs of nets.

The routed channel is stored in input-routed:SYMBOLIC (using oct data base) where "input" refers to the name of the input file. The routed channel can then be viewed using VEM. Presence of a capacitance library file with the name "cap\_library\_file" in the current directory is assumed. This file contains the coefficients used for calculating the coupling capacitances which can be automatically generated by the model generator CAP-MOD.

Expected syntax for the input file of ART is shown below. In this description (integer), (float) and (string) refer to entries in integer, floating point and string formats respectively.

"number\_of\_nets (integer)" - Number of nets in the channel.

"channel\_left (float)" - x coordinate of left edge of channel.

"channel\_right (float)" - x coordinate of right edge of channel

“number of top segments (integer)” - Number of top edges of channel (for irregular top boundary) immediately followed by the correct number of left,right positions and the heights (with respect to any arbitrary reference) of the top edges in the following format:

“x\_left x\_right y\_height (float) (float) (float)” - top edge coordinates.

“number of bottom segments (integer)” - Number of bottom edges of channel (for irregular bottom boundary) immediately followed by the correct number of left,right positions and the heights (with respect to the reference used for specifying top edges) of the bottom edges in the following format:

“x\_left x\_right y\_height (float) (float) (float)” - bottom edge coordinates.

“number\_of\_pins (integer)” - Number of pins on top and bottom of channel. Immediately after this line there should be the correct number of lines describing the net and coordinates of each pin in the following format:

“net\_name x-pos y-pos (string) (float) (float)” - pin information.

“left\_list (integer)” - Gives number of nets crossing left end of channel. This line should be immediately followed by:

“net\_name net\_name ... (string) (string) ...” - Any number of net names per line until the all the nets crossing the left edge have been specified.

“right\_list (integer)” - Gives number of nets crossing right end of channel. This line should be immediately followed by:

“net\_name net\_name ... (string) (string) ...” - Any number of net names per line until the all the nets crossing the right edge have been specified.

“net\_width (float)” - Subsequently declared nets have this width. Value applies until next net\_width statement.

“net\_width net\_name (float)” Sets width of given net only (creates new net if net\_name not already declared).

“net\_separation (float)” - Sets minimum separation between wires.

“contact\_overhang (float)” - Overhang beyond the wire.

“contact\_cut\_size (float)” - Minimum cut size for contact.

“layout\_resolution (float)” - Sets layout resolution. Layout in output is described in integers(oct units), where the unit value (in input file) = 1/layout\_resolution oct units.

“critical\_pairs (integer)” - Gives number of critical pairs in channel. This line should be immediately followed by correct number of lines giving information about each critical pair, in the following format:

“net\_name net\_name c\_bound d\_inf (string) (string) (float) (float)” - Names of nets having coupling constraint, followed by maximum allowed coupling capacitance (c\_bound) in fF and the influence distance (d\_inf) beyond which coupling is ignored.

“matched\_pairs (integer)” - Gives number of matched pairs in channel. This line should be immediately followed by correct number of lines describing the matched pairs in the following format:

“net\_name net\_name (string) (string)” - Pair of signal wires to be matched.

“shielding\_net (string)” - Specifies that this net is suitable for use as a shield. This line may be repeated.

blank line - Ignored.

line beginning with “!” - Comment line: ignored.

“read file\_name (string)” - Read text from specified file.

## OPTIONS

The options which can be used are as follows:

- w net-width - set default net width
- s net-separation - set default net separation
- r layout-resolution - set layout resolution
- o - Write result into OCT data base
- v - verbose (prints out messages while executing the program)

**COMPILATION**

For compilation, type “make” in first in each subdirectory of art/code except art/code/main. Then type “make” in art/code/main.

**EXAMPLES**

See “choudhur/art/examples” directory for examples. In the Eg1 subdirectory eg1 is the input file, in Eg2 directory eg2, and so on. Each subdirectory has a “cap\_library\_file”.

## Appendix C

# User's Manual of CAPMOD

### NAME

CAPMOD - an analytical model generator for interconnect capacitances

### SYNTAX

```
capmod [-r] [-o out_file] tech_file
```

### DESCRIPTION

CAPMOD generates analytical models for self and coupling capacitances associated with interconnects. It requires a "tech\_file" which contains the information about the technology. The technology is assumed to have two levels of interconnects. The first level is called METAL1 and the second level called METAL2. The analytical expressions of the capacitance models are stored by default in the file capmod.out in ASCII format. However, they can be stored in any other file using the "-o" option followed by the output file name. If "-r" option is used the program creates files containing "c" subroutines describing the models. These files can then be directly used in the capacitance extraction programs. The files which are formed are: capmodSELF.c (self capacitance routines), capmodCROSS.c (crossover capacitance routines), capmodPAR.c (routines for parallel lines on same layer), capmodPAR12\_nonoverlap.c (routines for nonoverlapping parallel lines in different layers), capmodPAR12\_overlap.c (routines for overlapping parallel lines in different layers).

All the capacitances are assumed to be in fF, and the parameters used in the analytical models are assumed to be in micron.

The `tech_file` should be of the following syntax:

- “`met1.thick (float)`”: thickness of METAL1.
- “`met2.thick (float)`”: thickness of METAL2.
- “`met1.min_width (float)`”: minimum width of METAL1.
- “`met1.max_width (float)`”: maximum width of METAL1.
- “`met2.min_width (float)`”: minimum width of METAL2.
- “`met2.max_width (float)`”: maximum width of METAL2.
- “`min_met1_sep (float)`”: minimum separation between two adjacent METAL1 lines.
- “`min_met2_sep (float)`”: minimum separation between two adjacent METAL2 lines.
- “`max_met1_sep (float)`”: maximum separation between two adjacent METAL1 lines.
- “`max_met2_sep (float)`”: maximum separation between two adjacent METAL2 lines.
- “`met1.layout_resolution (float)`”: layout resolution for METAL1 layer
- “`met2.layout_resolution (float)`”: layout resolution for METAL2 layer
- “`oxthick_below_met1 (float)`”: oxide thickness below METAL1.
- “`oxthick_between_met1met2 (float)`”: oxide thickness between METAL1 and METAL2.
- “`oxthick_above_met2 (float)`”: oxide thickness above METAL2.
- “`top_layer_er (float)`”: relative permittivity of the top layer above oxide.
- “`relative mesh size (float)`”: controls the value of the minimum mesh size (should be between 0 and 1 and preferably less than 0.25; smaller its value more is the accuracy, but larger is the CPU time).
- “`relative mesh ratio (float)`”: controls the value of the expansion ratio used between consecutive meshes (should be greater than but close to 1, preferably less than 1.5; smaller its value more is the accuracy, but larger is the CPU time).
- “`self_cap`”: line which indicates that models for self capacitance should be generated.
- “`cross_cap`”: line which indicates that models for crossover capacitance should be generated.
- “`parallel_cap`”: line which indicates that models for capacitances of parallel lines on same layer should be generated.
- “`parallel2_cap`”: line which indicates that models for capacitances of parallel lines on different layers should be generated.

## COMPILATION

For compilation, type “`make`” in the main directory.



**EXAMPLES**

Example technology file "tech\_file" is in "choudhur/capmod" directory.