The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits

by

Ron Yair Pinter

B.Sc., Technion, Israel Institute of Technology
(1975)

S.M., Massachusetts Institute of Technology
(1980)

Submitted in Partial Fulfillment
of the Requirements for the Degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

August 1982

© Massachusetts Institute of Technology 1982

Signature of Author Ron Pinter
Department of Electrical Engineering and Computer Science
August 30, 1982

Certified by Charles E. Leiserson
Charles E. Leiserson
Thesis Cosupervisor

Certified by Ronald L. Rivest
Ronald L. Rivest
Thesis Cosupervisor

Accepted by _____
Chairman, Department Committee

The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits

by

Ron Yair Pinter

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 1982 in partial fulfillment of the requirements for
the Degree of Doctor of Philosophy

Abstract

Programs for integrated circuit layout at the module assembly level are typically decomposed into two phases—placement and routing. In this thesis we investigate a third phase which is often implicitly assumed—layer assignment. This thesis studies how layer assignment methodologies interact with placement and routing.

A simple layer assignment methodology, which is also commonly occurring, is *river routing,* where all wires can be routed in a single layer. We give concise necessary and sufficient conditions for a channel to be river routable, and based on these conditions, give a linear-time algorithm to find the optimal placement of modules across a channel. We also show that determining whether wires can be river routed in an arbitrary polygon can be determined in linear time. We show that in a rectangular channel, the optimal decomposition of an arbitrary routing situation into the union of river routing situations can also be determined quickly. In addition, we give NP-completeness results for several other planar routing problems.

In addition, we define new characteristics of channel routing: *monotonicity* and *jogging,* and investigate their relation to minimizing the channel's width. In both the *Manhattan* and the *knock-knee* two-layer wiring models, U-shaped turns do not help to reduce width. On the other hand, jogging is essential to achieve this goal. We also generalize the definition of channels, and show that two-layer routability of certain configurations in T- and X-shaped channels can be checked in linear time.

Finally, we discuss optimal layer assignment as an "after the fact" consideration, when wiring is specified without layer designation. We give an $O(n^{2.5})$ algorithm for minimizing the number contacts required for two-layer realisations.

A glossary of layout problems is provided as an appendix.

Thesis Cosupervisors: Charles E. Leiserson, Assistant Professor of Computer Science and Engineering
Ronald L. Rivest, Associate Professor of Computer Science and Engineering

Key words. analysis of algorithms, computational geometry, design automation, graph theory, layout of integrated circuits, placement and routing, river routing, VLSI

Acknowledgements

I would like to thank the members of my thesis committee. Ron Rivest has provided support, guidance and encouragement throughout my career as a graduate student, for which I will be indebted and grateful forever. His judgment and technical advice were invaluable during critical stages of my work. Charles Leiserson found a way to channel my own efforts in a way that I was not aware of myself. The long hours I spent in his office debugging my own ideas and absorbing his acute observations made this thesis what it is. He also taught me how to write in *English* (as opposed to Pinterish). Gerry Sussman added a fascinating perspective to my research that made me aware of what I was doing in a broader context. All three were always there, ready to discuss whatever flaky ideas I had at first, saving me from pitfalls and telling me I could do it.

The research in Chapter III was done in collaboration with Charles Leiserson. Also, Jim Saxe made key contributions to the discovery of the algorithm in Section III.3, and the NP-completeness result in Section III.4 was obtained jointly with Mike Sipser. Alan Baratz deserves credit for the two-layer wiring model with overlaps and for his help in developing the examples of Section V.4.2.

When I came to MIT I had the fortune of meeting two senior graduate students in the Theory of Computation group, Andrea LaPaugh and Errol Lloyd. Their friendship, advice, and constant encouragement, during their tenure here and after they graduated, deserve my deepest appreciation.

No research can be performed in vacuum. The special atmosphere of Tech Square—the Eighth Floor and other parts of the building—is truly unique. I would like to thank all of the people with whom you could always hang around in the long hallways talking about everything from astronomy to blackjack. In particular, cheers to Alan Baratz, John Batali, John Mitchell, Flavio Rose, Dan Weise, and Rich Zippel.

I would also like to thank the organizations that contributed generously to my financial support while I was a graduate student. The Israeli-American Educational Foundation provided a Fulbright-Hayes travel grant for overseas travel. During the academic years of 1978/9 and 1979/80 I was supported by a Research Assistantship sponsored by the National Science Foundation under Grant No. MCS78-05849, which was supplemented by a scholarship from the Hebrew Technical Institute, N.Y., N.Y. In the academic year 1980/1 and the summer of 1982 my Research Assistantship was sponsored by the Defense Advanced Research Projects

Agency under Contract No. N00014-80-C-0622. For the academic year 1981/2 I was awarded an IBM Graduate Fellowship.

Last but not least comes my family. My parents' confidence in me in all circumstances brought me thus far. They taught me to value education and strive for excellence without sacrificing other things in life. My wife Shlomit always had the right things to say at the right time, although sometimes it took me a while how right she was. Her constant encouragement and drive helped me keep my own sanity, especially during the last long three months. Finally, little Noa has been a constant source of happiness and liveliness to all of us.

Note to the reader

Chapters I and VIII of this thesis address mostly methodological issues and can be read almost independently from the rest of the document. Chapter II contains the formal model required for the technical discussions that follow in Chapters III–VII. The results included in these five chapters can be divided into three topics: river routing (III, IV), channel routing (V, VI), and layer assignment (VII). Each topic can be read almost independently. The only exception is Section III.1 which is a prerequisite for Sections V.2.2 and V.4.1.

Chapter numbers always appear in roman numerals. Sections, figures, theorems, lemmas, and algorithms, are all numbered using arabic numerals, preceded by the chapter number. In the text, however, the chapter number prefix is dropped whenever the reference is to a figure (etc.) in the current chapter. In other words, chapter numbers are used only for cross referencing among chapters.

References are tagged by letters and numerals. For works of two or less authors the first syllables of their surnames are followed by the last two digits of the year of publication. If the number of authors is three or more, the first letters of their surnames constitute the letters part of the reference.

Contents

Abstract 2
Acknowledgements 3
Note to the reader 4
Table of Contents 5

Chapter I. Introduction 9
1. Layout methods 10
1.1 Layer assignment methods 10
1.2 Placement evaluation 10
1.3 Channel routing 11
1.4 Composition methodologies 11
1.5 Combining methodologies 11
2. Models and solution techniques 12
2.1 An example: the one-shift 13
2.2 Wiring models 14
2.3 Analysis tools 14
3. Thesis outline and major results 15

Chapter II. Discrete models for the layout problem 17
1. Modules and their placement 17
2. Routing and channels 18
3. Layer assignment 20
3.1 One layer models 20
3.2 Two layer models 20
3.3 Multi layer models 22

Chapter III. River routing in parallel channels 23
1. Necessary and sufficient condition for wirability 25
2. Interpreting routing conditions as placements constraints 29
3. A linear time algorithm for the placement problem in a single channel 34
4. The placement problem for multiple channels 38

5. Nonrectilinear and multi-layer wiring models 41
6. Extensions and conclusions 42

Chapter IV. Generalised river routing 47
1. Introduction 47
2. Placement 52
2.1 Cable routability 53
2.2 Flippable modules 56
2.3 Unflippable modules 58
3. Detailed routing 60
3.1 River routing in a simple rectilinear polygon 60
3.2 Routing around modules 74
4. Conclusions 75

Chapter V. Routing two-point nets across a channel 77
1. Terminology 78
2. Width 79
2.1 Minimizing width in conventional models (summary of results) 79
2.2 Minimizing layers and width in the via-free model 81
3. Monotonicity 84
4. Jogging 87
4.1 An optimal algorithm minimizing jogging for river routing 88
4.2 Excessive jogging may be required for other models 90
5. Resolution of vertical conflict cycles 92
6. Routing across a rectangle in arbitrary order 96
7. Conclusions 98

Chapter VI. Routing in and around junctions 99
1. Routing in T-shaped channels 100
1.1 Terminology and notation 102
1.2 Routing results 103
2. Pairwise ordering and routing in X-shaped channels 109
3. Other orderings (total, fixed, arbitrary) 112
4. Towards a methodology of propagating routing constraints 112

Chapter VII. Layer Assignment for interconnect 115
1. Background 115
2. Contact minimization 116
3. Extensions 123
4. Open problems 124

Chapter VIII. Discussion 125
1. From river routing to arbitrary interconnect 125
2. Design considerations 127
2.1 Cables and cable busses 127
2.2 Channel routing 129

2.3 Channel intersection and orderings 129
2.4 Global layer assignment 130
3. Routing versus routability: impact on placement 131
4. Conclusions 132

Appendix A. Glossary of problems 133

References 139

Biographical note 144

Chapter I

Introduction

Layout systems for digital integrated circuits traditionally decompose the layout problem into two subproblems: *placement* and *routing**. In this methodology, we first *place* predesigned pieces of the circuit — called *modules* — on the chip, and then *route* wires to interconnect common *signal nets.* Each module has *terminals* located along its boundary that must be interconnected properly using a given number of wiring layers which depends on the fabrication technology. The prime objective is to minimize the total area required to realize the circuit subject to various design rules (that ensure the feasibility of fabrication). This approach was used in the design of the "PI" system for placement and interconnect, developed here at MIT ([Bar81],[Riv82]).

The placement phase is devoted to assigning geometric positions and orientations to modules on the chip. Even when no routing is specified the problem of minimizing the chip area is NP-complete** [LaP80a]. The intractability of this phase is aggrevated when routing requirements are being taken into account.

During the routing phase, entire parts of the mask area are allocated to global signal interconnection. The task of routing wires within a specified area so as to achieve electrically valid connections of high quality constitutes a hard problem which is widely believed to be intractable on many counts. It is hard to optimize even one of several important layout characteristics, such as total area, wire length, and signal propagation delay. Even restricted versions of the problem, in which we isolate special subcases with a certain common structure, are still hard to deal with.

* For complicated VLSI circuits, one may need to form a hierarchy of such problems, as in [Pr79].

** See [GaJo79] for a discussion of NP-completeness.

Various design methodologies have been suggested to deal with the complexity of the layout problem, mostly within the placement-routing paradigm. We start by describing these methodologies and stating the contributions of this thesis to their development, application, and understanding of their interrelation. Next we discuss various models for studying the layout problem and justify our choice for modeling by a simple example. Finally we outline the thesis, highlighting its major results.

1. Layout Methods

Here we describe four methods for handling the awesome complexity of the layout problem. They operate on different levels, but are certainly relevant to each other in significant ways. The results of this thesis demonstrate how combinations of such techniques either lead towards efficient optimal solutions or are still intractable in spite of being seemingly simple.

1.1. Layer Assignment Methods

One method for dealing with the routing problem is to separate the geometric issue, which concerns area consumption and wire lengths, from the more involved problem of layer assignment, which affects mostly performance but also area. Various abstract models have been suggested to deal with the geometric aspect of the problem in such a way that layer assignment can be avoided altogether by fixing it in advance using simple global rules, such as Manhattan routing [Lec61]. Even in such a setting, most problems are proven or believed to be NP-complete ([LaP80a], [Sz81], [Ric81], [KvanL82] and others). Another result of this approach is that the quality of the layout produced in terms of the layer assignment may be extremely poor: wires may change layers where they do not need to or may consist of long runs of bad conductors. Moreover, due to their handling of design rules, these simplified models are *too weak* in the sense that certain features of the fabrication process are lost in the over simplistic restrictions on paths selection. For example, no overlaps between wires are allowed, although they may be extremely useful in reducing area. Paradoxically, the more general models sometimes give rise to tractable algorithms with provable properties regarding their output (*e.g.* [RBM81]); this situation calls for a better understanding of the layer assignment aspect of layout.

1.2. Placement Evaluation

Another issue is the interrelation between the placement and routing problems. The way in which modules are placed relative to each other may dramatically affect the difficulty of the routing problem and the quality of the solution. Ideally, we would solve the routing problem for each proposed placement (while looking for an optimal one), but this is obviously

intractable. Thus we need good estimates for the area needed to interconnect modules in a given placement without spending the time needed to actually solve the routing problem. Such estimates depend drastically on the layer assignment method assumed when these estimates are taken.

1.3. Channel Routing

One common method for overcoming the difficulties inherent to the global nature of the routing problem was suggested by Hashimoto and Stevens in their seminal paper on *channel routing* [HaSt71]. They proposed to partition the routing area into rectangular *channels* which are then routed individually. This breaks the original problem into a set of smaller, local problems, thus reducing the complexity of their solution. The major disadvantage of this approach is the fact that once the subproblems are set up, their solutions do not interact, and global patterns are lost in the process. In other words, the decomposition into subproblems is *geometric* and has hardly any bearing on the interconnection pattern that is set up by the terminal nets.

1.4. Composition Methodologies

Another way of managing the layout problem is implied by various restricted design methodologies, according to which modules can be put together only in certain fashion, thus giving rise to specific routing situations. Prime examples of layout systems purporting such methodologies are Bristle-Blocks [Jo79,Jo81], the data-path generator of DPL/Daedalus [BMSSW81,Sh82], MacPitts [SSC82], and Ali [LSV82,LNSVV82]. Although their main objective is to provide a tool for specifying one's layout in functional terms (stressing a structured style), a beneficial side effect is a considerable simplification of the routing problem. However, the impact of such methodologies on layout algorithms has been by and large ignored by practitioners and theoreticians alike.

1.5. Combining Methodologies

In this thesis we investigate the impact of combining several of the above mentioned methodologies on the quality and efficiency of layout algorithms. First and foremost comes the layer assignment issue: how do different layer assignment strategies affect the power and complexity of solutions in the various contexts set up by each of the other methodologies? For instance, how are channel routing algorithms affected by the selection of a layer assignment method? We also investigate the effects of layer assignment on combinations of methodologies, e.g. how a one-layer strategy affects the placement evaluation in a data-path assembly system.

Another major concern is to find succinct routability conditions and how they affect placement considerations both in the general setting (of arbitrarily composed rectangular modules) and in the restricted composition methodologies. In the first case, attention focuses on phenomena occurring due to interaction between conventional channels. Here we suggest that the routing situation inside each channel be inspected, and from this deduce information about the interaction between neighbouring channels. Only then can one make decisions concerning the details of the wiring. In the second case, when restricted composition methodologies are applied, we demonstrate how certain composition rules lead to routability conditions that are sufficiently well-behaved to facilitate accurate and discrete modeling of the entire layout problem, which then becomes amenable to optimal and efficient algorithmic solutions.

The question of optimal layer assignment by itself is also examined here in a somewhat independent manner: assuming the layout has been completed somehow, we are interested in algorithms to assign layers after the fact. Naturally, the feasibility and quality of the resulting circuits still depends on the tools used initially to devise the geometry.

Finally, this thesis also includes an investigation into fundamental problems concerning channel routing strategies. Layer assignment does not have a direct impact on these considerations, neither have they an influence on placement evaluation, but these results in themselves comprise methodological guidance in themselves.

2. Models and Solution Techniques

The methods be employed in this research are intentionally discrete, with emphasis on exact analysis of performance, both in terms of the optimality of the results and the complexity of the algorithms used. To put this goal in perspective, a brief review is due. Over the past two decades of research on layout, three major schools have emerged. One school, which comprises Akers [Ak72], Donath [Do79], Heller *et.al.* [HMD78], Soukup and Royle [SouRo81], and subsequently El-Gamal [ElG81], has engaged in continuous analysis of routing phenomena and their effect on placement considerations. They try to model the effects of different terminal distributions using real-valued random variables, and thus obtain good estimates on routing requirements and the interaction between routing situations. The heuristics guiding detailed routing are influenced by this modeling and do not pay much attention to local phenomena.

The second approach is primarily represented by Kuh and his students (*e.g.* [YoKu82] and [MaKu82]) and by the work on the LTX system at Bell Laboratories [Deu76], [PDS77]. Here discrete methods are used to try and capture the fine points that are due to local

perturbations in the terminals' set-up. These methods have been rather successful in practice, but no attempt has been made to derive any formal results on their performance, especially when it comes to the relation between their solutions and the optimum.

Recently, the theoretical computer science community started to develop interest in this area, and some careful analyses have been worked out that show how trivial changes to the routing configurations may cause disastrous effects on the quality of the layout. A few of the early works that comprise this third school are [LaP80a], [DKSSU81], and [BrRi81]. These recent results and the advances in the design and analysis of algorithms over the past decade call for a careful examination of the layout problem using discrete tools.

I believe strongly that despite the apparent difficulties, discrete analysis is the correct method of discourse to deal with the layout problem. It is a mistake to use the wrong methods because they seem to work out easily; problems that are discrete in nature have to be dealt appropriately to yield the right kind of results.

2.1. An Example: the One-shift

A nice example that brings this point home is the *one-shift* channel (Figure 1): imagine two identical rows of n terminals that must be connected in order. This goal can be achieved without any cost in area or wire length by simply putting the two rows right next to each other. If, however, you place them on modules across a channel such that one row is shifted against the other by just *one* grid unit*, you are in for an unpleasant surprise. Depending on how many layers you are allowed to use and on the fashion in which you are allowed to route them, you may have to place the two modules n grid units apart (for one layer realization [DKSSU81], [LeiPi81]), $O(\sqrt{n})$ units apart (for Manhattan routing [BrRi81]), or just one unit apart (for "real" two-layer design rules [Bar81],[LeiPi81]). Thus small perturbations in placement create vastly different routing situations, incurring area penalties that are orders of magnitude apart.

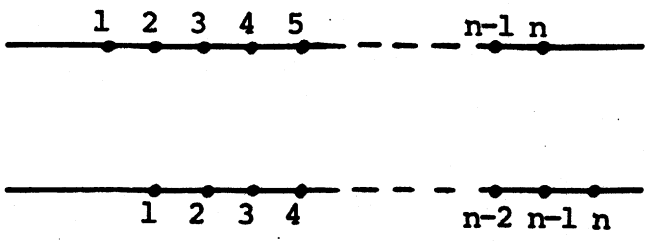


Figure I.1: The *one-shift* channel routing problem: *large* width (separation) may be required due to the *slight* misalignment of terminals.

* We assume an underlying grid whose unit corresponds to the minimum center-to-center separation required to layout adjacent wires. For simplicity, at this stage, we also assume that this separation is uniform for all layers. Formal models for wiring are exposed in Chapter II.

2.2. Wiring Models

The discrete modeling of wiring plays a key role in faithfully representing the phenomenon of the one-shift example. Various models give rise to widely different results. Different models reflect different layer assignment methods, and it is our task to match models with practice. Another issue concerning modeling is the relation to actual mask data and design rules. Obviously, a model could be cluttered with excessive amounts of irrelevant detail, but on the other hand, making it too simple may render its results irrelevant. It is our goal to come up with a model that yields to efficient algorithmic methods and reflects the essential parts of the original problem at the same time. Also, a good model is one that can be *naturally* extended to represent the additional details if necessary.

2.3. Analysis Tools

Tools of discrete mathematics are used throughout this thesis. Most problems are solved by using techniques of graph theory and computational geometry. As much as possible, we try to characterize problems in an exact manner within the framework established by these disciplines.

Mostly, we strive to find *optimal* solutions that can be computed efficiently (*i.e.* in polynomial time). In other cases we can prove that specific configurations give rise to NP-complete problems. In some cases, however, we cannot prove completeness for some difficult class, but we still do not know of efficient algorithms to solve them optimally. Then one has to resort to heuristics, which may not be as accurately analyzable as optimal algorithms, but should be discrete in their modeling of the problem.

Once (desirably accurate) estimates for routing requirements are found and expressed as succinct discrete structures, we can manipulate them abstractly in order to make global routing decisions. A novel approach suggested here is to apply heuristics to the intermediate abstractions provided by the first, algorithmic stages, rather than to try out heuristics from scratch. Furthermore, the heuristics could be borrowed from an entirely different domain than combinatorial optimization, and hence their evaluation will have a different flavor.

One technique that could be used is that of *constraint propagation,* as exposed in [StSu79] and [St80]. Some modifications to the methods proposed in Steele's thesis [St80] are due in order to make them applicable to nonscalar entities. Since our constraints will relate algebraic objects such as relations and aggregates, and the constraints themselves may involve maximization and minimizations, this task is nontrivial and is expected to require some interesting extensions to [St80] which are beyond the scope of this thesis.

3. Thesis Outline and Major Results

Chapter II introduces the models used throughout this thesis* to represent the layout problem and layer assignment methods. The first set of results to be presented applies to a simple layer assignment strategy, namely — routing in only *one* layer. In Chapter III we discuss the problem of *river-routing* across a channel. We show how accurate routability conditions can be derived for a variety of one-layer wiring models, and how those conditions can be used to solve the placement problem for river routing optimally in linear time for the rectilinear model. The problem, however, becomes intractable once multiple channels, as might be encountered in a data path, are considered.

From this relatively simple configuration of river routing across a channel, we extract a few essential properties that prove to be interesting from a methodological point of view as well as rich in algorithmic content. Chapter IV takes up the problem of *planar* routability and its impact on placement. The line between "easy" and "hard" problems is thin. Planar routing in a simple polygon is tractable, but once holes are introduced they render the problem NP-complete. The topological issues concerning placement are shown to be easy, which is not surprising considering the trivial topology of the interconnect.

In order to handle more complicated situations one must use stronger strategies — both for layer assignment and for dealing with the interaction between routing areas. In Chapter V we go back to the channel context, looking at routing two-point nets *across* a channel. Various routing strategies and their impact on minimizing the channel width are investigated. In Chapter VI the channel structure is generalised, and routing patterns in T-shaped and X-shaped channels are investigated. Some tractable situations are identified. Then, based on the observations that lead to efficient algorithms, we develop a methodology that suggests ways to combine many small solutions to solve the general routing problem in a coherent way.

In Chapter VII, we turn around to ask how layers could be assigned retroactively so as to make a given piece of artwork realizable within certain design rules. A polynomial-time algorithm is provided to solve the two-layers assignment problem for a variety of optimization criteria, disclaiming a previous NP-completeness conjecture. We also set the ground for a more careful study of the subject in a graph-theoretic framework.

Finally, in Chapter VIII we reflect on the results presented in this thesis, trying to draw the lessons that both a circuit designer and the purveyor of design aids should learn from our investigations. For readers who are interested in the combinatorial complexity aspects of this thesis, a glossary of problems and their status (in the form of [GaJo79]) is provided in Appendix A.

* The terminology defined in Chapter II may be changed locally in some of the subsequent chapters as indicated clearly in their introductions.

Chapter II

Discrete Models for the Layout Problem

Throughout this thesis, we assume that layout is generated over an underlying *square grid*. A point in the plane is a *grid-point* if both its coordinates are integers. A line in the plane of the form $x = m$, where m is an integer, is a vertical *grid-line*. Likewise, a line of the form $y = m$ is a horizontal grid line.

1. Modules and Their Placement

A *module* is a rectilinear polygon, *i.e.* its sides are either parallel or perpendicular to each other; furthermore, the dimensions of all sides are integral. Also, a set of distinct points, called *terminals*, is associated with each module such that each terminal lies on one of the module's sides at an integral distance from (either of) its end(s). Each terminal is usually labeled by the name of a *signal-net*. Thus, if we pick an arbitrary corner* of a module and align the two sides meeting at the corner with a horizontal grid-line and a vertical grid-line, all terminals and other corners of the module will fall at grid points, and all sides will coincide with segments of either vertical or horizontal grid-lines.

A *placement* of a set of modules is an assignment of each module to a portion of the grid such that its corners coincide with grid points and its sides — with grid lines, and no two (areas covered by) modules overlap. Furthermore, no two sides of modules or their corners may coincide. Thus the distance between any two points within or on the boundary of two modules is at least 1. Figure 1 shows a legal placement for a set of rectangular modules on a grid.

* *i.e.* a point along a module's boundary where two perpendicular sides meet.

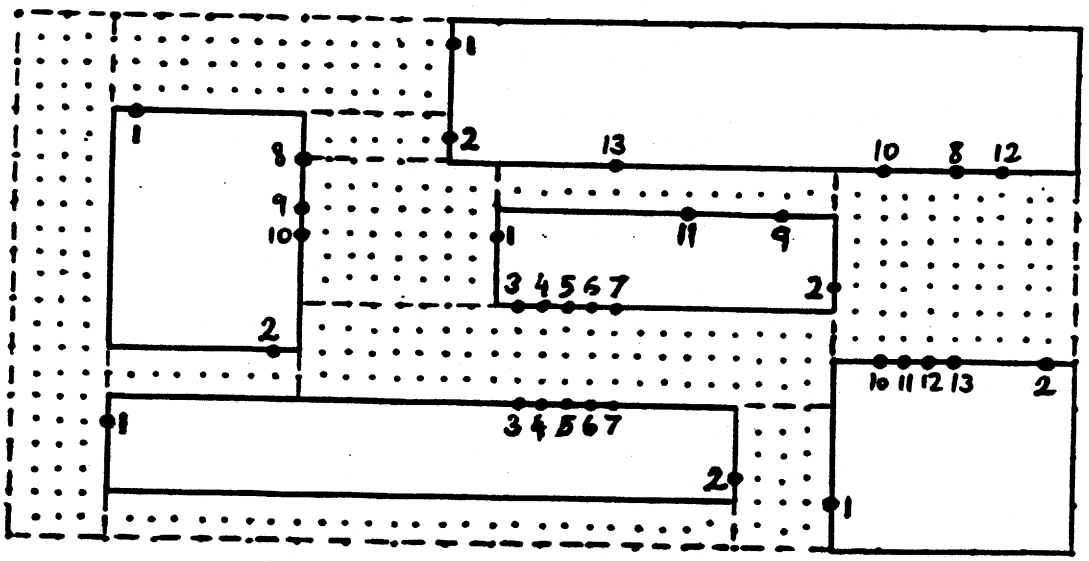


Figure II.1: A placement of five rectangular modules on the grid. The dashed lines show a possible partitioning into channels.

The orientation of a module in a placement may be constrained in various ways, depending on the application domain and methodological considerations. The operations that can be performed on a module before placing it are (rectilinear) rotation and reflection. *Rotation* allows us to turn the module by 90°, 180°, or 270°. *Reflection* allows us to mirror a module — with respect to to the x-axis or the y-axis. These operations are realisable at the mask level for integrated circuits without affecting any of the physical aspects of the design; reflection, however, may not be as easy to realise in a PCB design.

2. Routing and Channels

A placement of modules on the grid defines the locations of terminals that have to be connected. *Wires* that serve to realise the interconnect are modeled by paths on the grid. A *path* between two grid points, P and Q, is a sequence of grid points $R_1, R_2, \ldots, R_k$ such that $P = R_1$, $Q = R_k$, and each two successive points, R_i, R_{i+1} for $i = 1, \ldots, k-1$, lie on the same grid line. These line segments (including their end points) are called the path's *segments* and are denoted by $[R_i, R_{i+1}]$ for $i = 1, \ldots, k-1$. Furthermore, the points R_i for $i = 2, \ldots, k-1$ are called *turning points*, and we insist that at each turning point a vertical segment meets a horisontal one.

Generally, wires are not allowed to go through modules or lie on their sides. Thus routing is completely disjoint from the modules, and the segment attached to each terminal is perpendicular to the module's side (going away from it). Sometimes, for purposes of notational

convenience (especially in river routing situations), we allow one side to be free for routing, and we shall make exceptions to our definitions as needed.

The routing area can be subdivided into *channels*. Channels are rectilinear, non–overlapping polygons with corners and sides conforming with the grid (as modules do). Channels may share sides (and corners) with each other and with modules. In Figure 1, the routing area has been partitioned into (rectangular) channels as shown by the dashed lines. A terminal belonging to a module whose side is shared with a channel is now associated with that channel's side. Obviously, the channels and modules do not cover the whole (infinite) grid; the objective of the layout problem is usually to minimize the area of the smallest rectangle enclosing the ensemble (sometimes called the *bounding box*).

Traditionally, channels are rectangles. We shall deviate from this convention in Chapter VI, but until then some terminology particular to the common rectangular structure is needed. In a rectangular channel, each horizontal grid line lying within the channel is called a *track*. Segments of vertical lines bounded by the sides are called *columns*. The *width* of a channel is one more than the number of tracks, and, likewise, its *length* is one more than the number of columns. A horizontal path segment is called a *jog track* and a vertical one — a *jog column*.

In some contexts, a rectangular channel does not have fixed dimensions—only the shape of a rectangle has to be maintained. Routing is confined to the infinite stripe between two parallel lines, as can be seen in Figure 2. The two sides, the top and the bottom, have fixed lateral positions, but they can move up and down. The width of the channel is the distance between the top and bottom sides (and is still one more than the number of tracks), and the channel's length is the difference between the x-coordinates of the leftmost and rightmost columns used for routing.

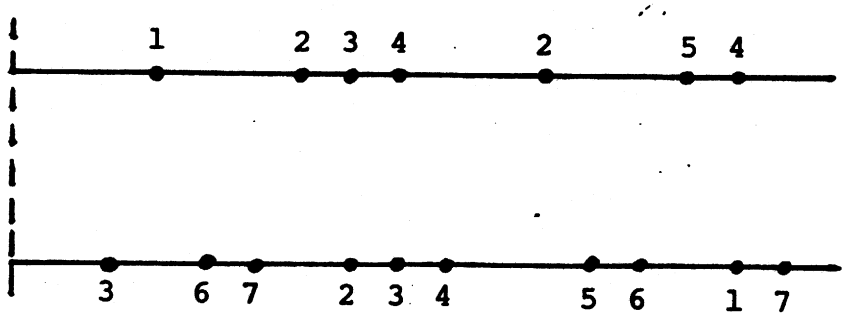


Figure II.2: Routing across a rectangular channel.

3. Layer Assignment

In order to realize the wires, layers have to be assigned to the paths representing them. A set of rules that specifies how paths can be run on the grid so that layer assignment becomes feasible is called a *wiring model.* Note that although the purpose of devising such rules is to facilitate proper realization with respect to layers, the rules are formulated entirely in terms of paths. Thus wiring models provide a powerful abstraction that serves to reduce the complexity of routing problems.

Wiring models are classified by the number of layers they use. There are, however, a few models that belong to more than one such class, as we note throughout the description of the models.

3.1. One-layer Models

To model one-layer realization, paths corresponding to different wires must be distinct: two different paths may not have any overlapping segments, all of their turning points have to be disjoint, and no point of one path is allowed to lie on segments of the other. This wiring scheme abstracts a uniform unit separation design rule, and is appropriately called the *(one-layer) square grid* model. Figure 3 shows a typical routing in this model. In general, one-layer models can be used only when the interconnection pattern is planar. In the channel case, for example, the order of the terminals' labels has to be identical on both sides, to avoid crossovers. For a channel, such a situation is called *river routing.*

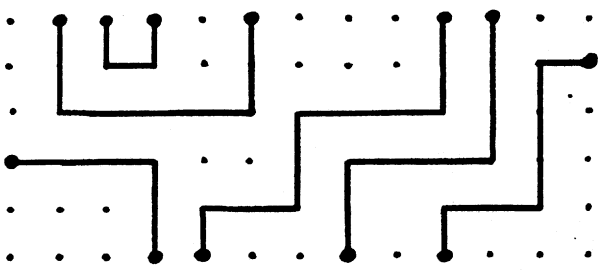
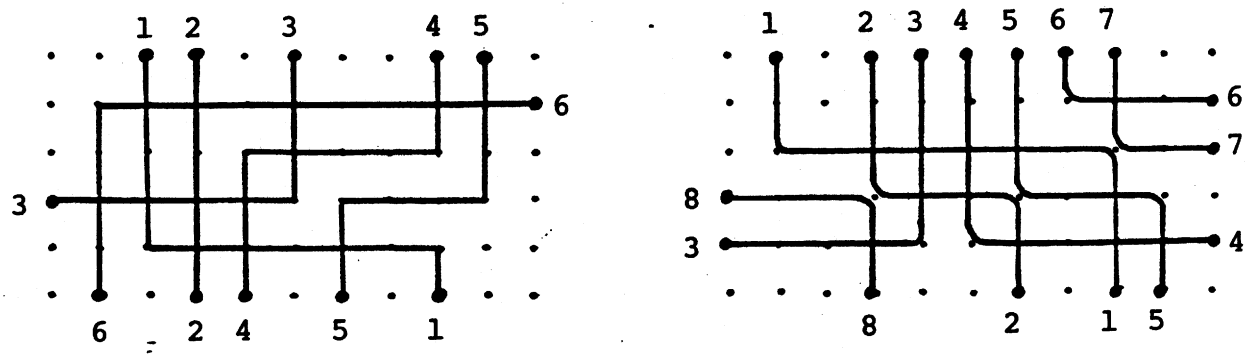


Figure II.3: Routing in the one-layer model.

3.2. Two-layer Models

For two-layer realizations we have a variety of models, each reflecting certain methodological considerations. First we have the *directional* model: each direction in the grid is preassigned to one of the two layers. Thus, for any two paths no turning point of one path may

lie on a segment of the other. In other words, the paths may cross each other, but cannot share segments, or turn at the same grid–point. This scheme conforms with the traditional Manhattan wiring model [Lee61]. In current technology (*e.g.* nMOS, see [MeaCo80, Chapter 2]), connections between the layers are facilitated by *contacts* (sometimes called *vias*), hence the justification for not making two turns at the same point (causing two contacts to overlap). The directional model is illustrated in Figure 4(a).



(a) The directional model (b) The knock-knee model

Figure II.4: Legal routing in Two-layer models.

Second, we have the *knock-knee* model, which was introduced by Thompson [Th80]. Here paths are allowed to share corners, *i.e.* they may have common turning points, but no segment overlap is allowed. Layers must be assigned in such a way that whenever a corner is shared between two paths (wires), they are in different layers. This model has the advantage of avoiding undesirable electrical properties that are due to overlaps between different layers, such as capacitive coupling (see, for example, [MeaCo80, Section 1.4]), but the complexity of layer assignment may become the overriding issue since now we have the freedom to change layers at any "free" grid-point. This subject is discussed in detail in Chapter VII. An example of a legal routing in this model is shown in Figure 4(b).

Finally, we have the general model which differs from actual routing with two layers only by imposing a uniform unit separation design rule. For completeness sake we present here a definition of the model (developed jointly with Alan Baratz), but give no further analysis. At each grid point, the following rule must be satisfied. If the point contains a contact, then only one path goes through it and its layer gets changed there; if no contact is present, up to two paths may go through the point in any way they please.

3.3. Multilayer Models

The next model for routing in two or more layers takes a different approach to layer assignment altogether. In the *via-free* model, each net (wire) is routed in one layer and may not change layers along the path. We assume that all terminals are available in all layers, so each net can be assigned a layer by the routing algorithm. This approach alleviates the problems caused by using contacts, at the expense of not being a "general" wiring model. Figure 5 shows that not every interconnection pattern can be realized using the via-free wiring model, but it can still be used to embed many useful networks, such as the shuffle-exchange graph. This "one layer per net" model is in many ways a natural generalization of one-layer routing, as we shall see in the results of Chapters III and V.

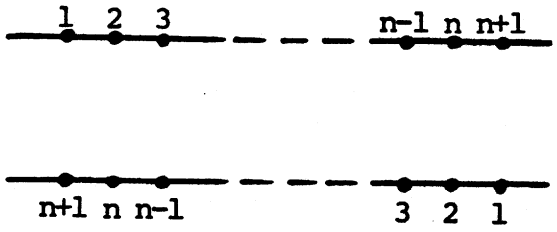


Figure II.5: A channel that cannot be routed in the via-free model using n layers.

General models for routing in more than two layers involve complicated rules concerening contacts: A contact between layer i and layer j may not coincide with any wiring on the layers k in between, $i < k < j$. Providing a specific wiring model for three layers may still make sense, but for four layers or more we end up, in fact, routing in a three-dimensional medium [Ro81]. This domain is beyond the scope of this thesis.

Chapter III

River Routing in Parallel Channels

River routing across a channels is a special routing problem which arises often in the design of integrated circuits, and it has been shown to be optimally solvable in polynomial-time for many wiring models (see in particular [Bar81], [DKSSU81], [SieDo81], and [To80]). In this chapter, we improve the formulation of *routability* conditions for one channel, and demonstrate that the placement problem for river routing across one channel is also polynomial-time solvable. Further, we explore the situation of river routing in several *parallel* channels and draw a fine line between tractable and intractable placement problems in this situation.

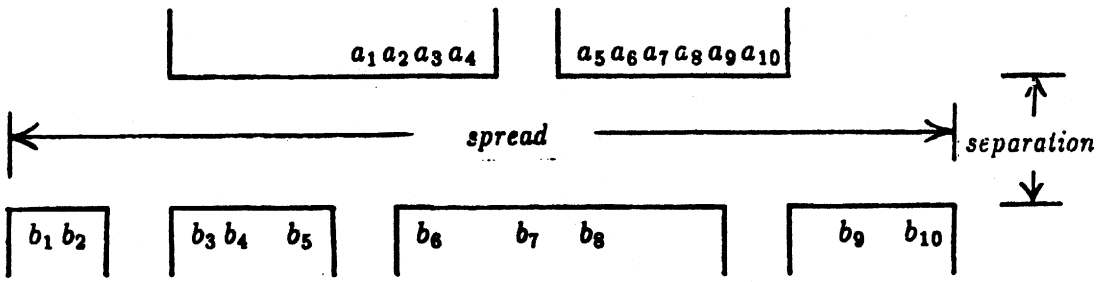


Figure III.1: Two sets of chunks on either side of a rectangular channel. Terminal a_i must be connected to b_i for $i = 1, \ldots, 10$.

The general character of the placement problem for river routing is illustrated in Figure 1. Two sets of terminals $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ are to be connected by wires across a rectangular channel so that wire i is routed from a_i to b_i. The terminals on each side of the channel are grouped into *chunks* which must be placed as a unit. The quality of a legal placement—one for which the channel can be routed—can be measured in terms of the

dimensions of the channel. The *separation* is the vertical distance between the two lines of terminals, and the *spread* is the horizontal dimension of the channel.

The *wiring model* gives the constraints that the routing must satisfy. Although our results can be generalized to include a variety of wiring models (see Section 5), we concentrate on the *(one-layer) square-grid model.* Recall that crossovers are disallowed in the square-grid model, and all wires must take disjoint paths through the grid.

The placement problem for river routing arises often during ordinary integrated circuit design. A common instance is when the terminals of one or more modules are to be connected to drivers. The various independent "chunks" are the modules, which lie on one side of the channel, and the drivers, which lie on the other.

A more interesting manifestation of the placement problem occurs in the context of design systems such as bristle-blocks [Jo79,Jo81] and DPL/Daedalus [BMSSW81,Sh82]. These systems encourage a designer to build plug-together modules so that the difficulties associated with general routing can be avoided. A designer may specify *stretch lines* which run through a module and allow the module to be expanded perpendicular to the stretch line, as demonstrated in Figure 2. When two independently designed modules are plugged together, stretch lines permit the terminals to be *pitch aligned*, that is, the distances between pairs of adjacent terminals are made to match the distances between their mates, and routing is avoided because the separation of the channel is zero. Unfortunately, this approach may not succeed unless stretch lines are put between every pair of adjacent terminals. The stretch lines may not only disrupt the internal structure of the modules, but the consequence may be an inordinate amount of stretching that leaves the channel with a large spread.

The other extreme is to forego stretching altogether and river route between the terminals. But the cost may still be large if a large separation is required in order to achieve a routing. A reasonable compromise is to place stretch lines where it is convenient, and then do a little stretching and a little routing. Determining how much of each to do is exactly the placement problem for river routing.

Most of the results reported in this chapter repersent joint research with Charles E. Leiserson. Section 1 gives a concise necessary and sufficient condition for a channel to be routable in the square-grid model. Section 2 shows that the form of this condition allows the placement problem to be reduced to the graph-theoretic problem of finding the longest paths from a source vertex to all other vertices in a graph. Based on this problem reduction, a linear-time algorithm for optimal placement is given in Section 3. The discovery of a linear-time algorithm was initiated by discussions with James B. Saxe of Carnegie-Mellon University. Section 4 shows that the placement problem for multiple, parallel channels is NP-

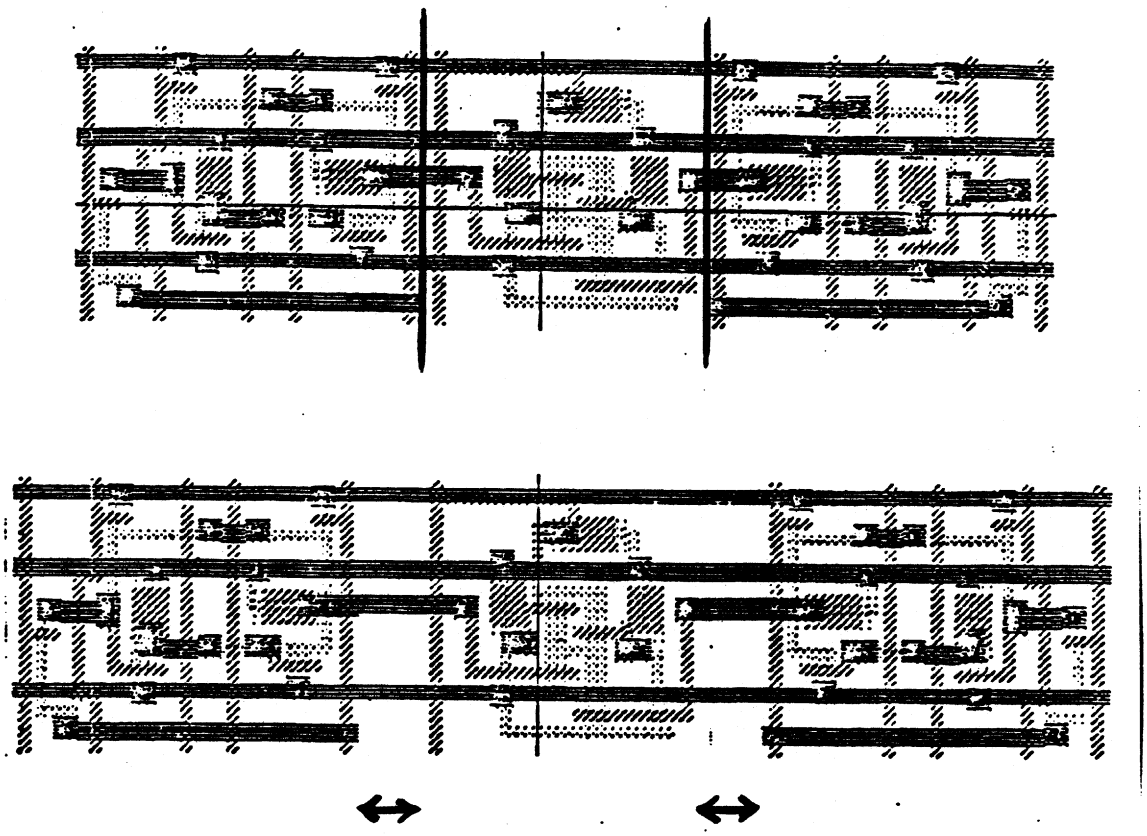


Figure III.2: A module before and after stretching (courtesy of John Batali).

complete if the individual channel widths are not specified, but an algorithm whose running time is exponential in the number of *channels* alone can be used to find an optimal solution. The NP-completeness proof represents joint research with Michael Sipser. Section 5 shows that the placement algorithm for a single channel extends to wiring models other than the square-grid model, but its performance depends on the particular wirability conditions for the model. Section 6 discusses the application of our results to other routing situations and suggests further placement problems.

1. Necessary and Sufficient Conditions for Wirability.

Figure 3 shows a solution to the problem of Figure 1 using the square-grid model. As the figure suggests, wires have width and minimum spacing between them. Throughout this chapter (alone), we adopt the convention that a grid point corresponds to the lower left portion of a wire. In our model the terminals $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ occupy grid points on opposite sides of the channel. As can be seen in the figure, we obey the lower-left convention by routing wires on the bottom row of grid points in the channel but not on the top row. This convention

also allows terminals to be located at the left corner of a chunk, such as b_6, but not at the right corner.

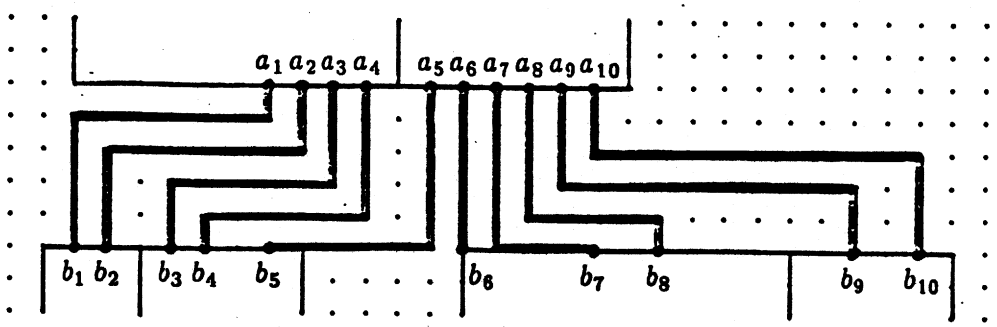


Figure III.3: A possible solution to the problem in Figure 1 for which the separation is 5 and the spread is 28.

In order to establish constraints on wirability in this model, consider a straight line segment drawn from (x_1, y_1) to, but not including, (x_2, y_2). We ask the question, "How many wires can cross this line?" With a simple analysis we can show that the answer is $\max(|x_2 - x_1|, |y_2 - y_1|)$. Without loss of generality, assume the situation is as in Figure 4, and look at the grid points immediately below the line, that is,

$$\left\{(x, y) \mid x_1 \leq x < x_2 \text{ and } y = \left\lfloor y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \right\rfloor \right\}.$$

Any wire crossing the line must perforce occupy one of these grid points, and therefore the number of such wires is bounded by the cardinality of this set.

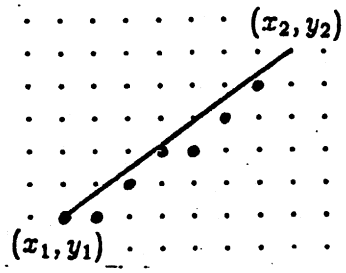


Figure III.4: The number of wires crossing the half-open line segment is at most the number of grid points immediately below the line.

Let us now turn to the river routing problem and examine how this constraint can be brought to bear. Let $a_1, \ldots, a_n$ denote both the names of the terminals at the top of the channel and their x-coordinates, and let the same convention hold for the terminals $b_1, \ldots, b_n$ at the bottom of the channel. Figure 5 shows a half-open line segment drawn from terminal b_i to the grid point immediately to the right of terminal a_j. The $j - i + 1$ wires emanating from $a_i, \ldots, a_j$ must all cross this line. Similarly, the $j - i + 1$ wires emanating from $b_i, \ldots, b_j$

must all cross a line drawn from a_i to $b_j + 1$. In order for a channel with separation t to be routable, therefore, it must be the case that

$$\max(a_j - b_i + 1, t) \geq j - i + 1 \quad \text{and} \quad \max(b_j - a_i + 1, t) \geq j - i + 1 \tag{1}$$

for $1 \leq i \leq j \leq n$.

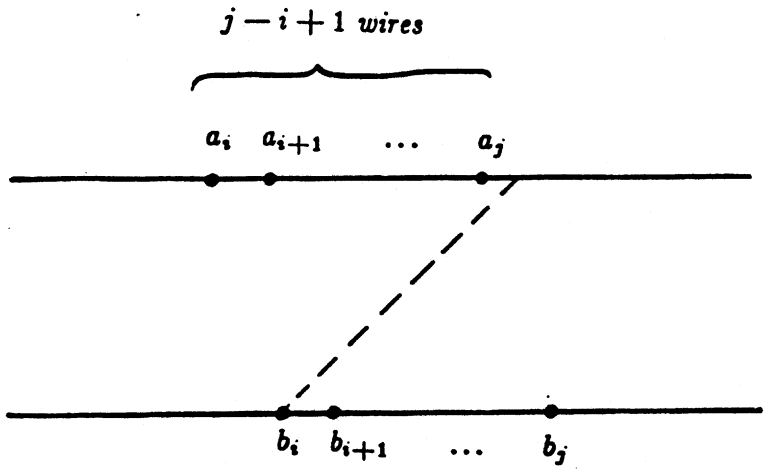


Figure III.5: The $j - i + 1$ wires from $a_i, \ldots, a_j$ must cross the dashed line between b_i and a_j.

Although Condition (1) is a new condition for wirability, the analysis that leads to it is essentially the same as that in [DKSSU81] and represents previous work in the field. One of the contributions of this chapter is to provide a more compact condition which is equivalent:

$$a_{i+t} - b_i \geq t \quad \text{and} \quad b_{i+t} - a_i \geq t \tag{2}$$

for $1 \leq i \leq n - t$. The channel is always routable if $t \geq n$.

Condition (1) implies Condition (2) because Condition (2) can be obtained by substituting $j = i + t$ in Condition (1). For the opposite direction, suppose first that $j - i + 1 < t$; then $\max(a_j - b_i + 1, t) \geq t > j - i + 1$. If $j - i + 1 \geq t$, on the other hand, then

$$\begin{aligned} a_j - b_i + 1 &= a_{i+t+(j-i-t)} - b_i + 1 \\ &\geq a_{i+t} - b_i + 1 + (j - i - t) \\ &\geq t + 1 + (j - i - t) \\ &= j - i + 1 \end{aligned}$$

since $a_{k+1} \geq a_k + 1$ for all $1 \leq k < n$. Thus the two conditions are indeed equivalent.

Figure 6 shows a simple geometric interpretation of Condition (2). The condition $a_{i+t} - b_i \geq t$ means that a line with unit slope going up and to the right from b_i must intersect the top of the channel at or to the left of terminal a_{i+t}. If the condition fails, terminal b_j must

be to the right of a_j for $i \leq j < i+t-1$, that is, each wire from a a_j goes down and to the right, which can be shown to follow from the fact that $a_{j+1} \geq a_j + 1$. Thus the geometric interpretation of failure is that too many wires are trying to cross the line of unit slope. (For $b_{i+t} - a_i \geq t$ the line with slope -1 going down and to the right from a_i must intersect the bottom of the channel at or to the left of terminal b_{i+t}.)

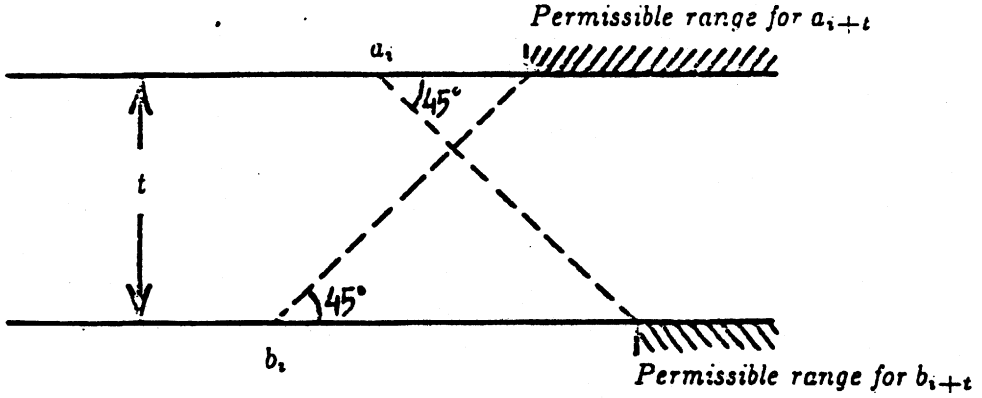


Figure III.6: Geometric interpretation of $a_{i+t} \geq b_i + t$ and $b_{i+t} \geq a_i + t$.

This geometric interpretation can be used to show that Condition (2) is not only a necessary condition for routability of the channel, but a sufficient condition as well. In fact, a simple greedy algorithm will successfully route a routable channel. Processing terminals left to right, the greedy algorithm routes each wire across the channel until it hits a previously routed wire; then it follows the contour of the opposite side until it reaches its destination.

To see that this algorithm works given Condition (2), we must be more precise about what paths are taken by the wires. Consider without loss of generality a block of consecutive wires that go down and to the right, that is, $a_i \leq b_i$ for all wires in the block. For any horizontal position x such that $a_i - t < x \leq b_i$, define

$$\eta_i(x) = \max(a_i - x, \max_{b_{i-r} \geq x} r).$$

The path of wire i is then described by the locus of points $(x + \eta_i(x), \eta_i(x))$ for $a_i - t < x \leq b_i$.

A geometric interpretation of this formulation uses the same intuition as was given in Figure 6. The line with unit slope drawn from $(x, 0)$ where x is in the range $a_i - t < x \leq b_i$ must cross wire i. The value $\eta_i(x)$ gives the y-coordinate of wire i where it crosses this line of unit slope. The two-part maximum in the definition of $\eta_i(x)$ corresponds to whether the wire is being routed straight across the channel or whether it is following the contour of the bottom. The value of $\eta_i(x)$ for the latter situation is the number of wires to the left of wire i which must cross the line of unit slope.

We must now show that the locus of points for a wire is a path, that the paths are disjoint, and that they never leave the channel. That the locus of points is indeed a path can be seen by observing that as x ranges from $a_i - t$ to b_i, the initial point is $(a_i, t-1)$, the final point is $(b_i, 0)$, and with a change of one in x the coordinates of the path change by a single grid unit in exactly one of the two dimensions. To show that the paths are disjoint, consider two adjacent wires i and $i+1$, and observe for $a_{i+1} - t < x \leq b_i$ that $a_i - x < a_{i+1} - x$ and $\max_{b_i - r \geq x} r < \max_{b_{i+1} - r \geq x} r$, and therefore $\eta_i(x) < \eta_{i+1}(x)$.

To show a path of a wire never leaves the channel, we demonstrate that $\eta_i(x) < t$ for all i and x in the associated range. It is for this part of the proof that we need the assumption that Condition (2) holds. If for a wire i, the two-part maximum in the definition of $\eta_i(x)$ is achieved by $a_i - x$, then $\eta_i(x)$ must be less than t because $x > a_i - t$. Suppose then, that the two-part maximum is achieved by the maximal r such that $b_{i-r} \geq x$. To show that $r < t$, we assume the contrary and obtain a contradiction. But since $b_{i-t} \geq b_{i-r} \geq x > a_i - t$, the contradiction is immediate because $a_i - b_{i-t} \geq t$ from Condition (2).

2. The Structure of the Placement Problem.

The objective of a placement algorithm is to set up a routing problem that is solvable and minimizes some cost function. Many criteria can be adopted to measure the cost of a placement for river routing, whether in terms of area (total or channel) or some other function of spread and separation. A plot of minimal spread versus given separation reveals that the region of feasible placements may not be convex although the curve is guaranteed to be monotonically decreasing. (Figure 7 shows the plot for the problem of Figure 1.) Any measure of placement cost that is a function of spread and separation and which is monotonically increasing in each of spread and separation will therefore find a minimum on this curve.

Thus we content ourselves with producing points on this curve, that is, *determining a placement which achieves the minimum spread for a given separation t,* if indeed the channel is routable in t tracks. If minimum separation is the goal, for example, binary search can determine the optimum t in $O(\lg t)$ steps. Since the algorithm presented in the next section determines a placement for fixed t in $O(n)$ time where n is the number of terminals, and since the separation need never be more than n, a minimum-separation placement can be achieved in $O(n \lg n)$ time. For more general objective functions such as area, the optimal value can be determined in $O(n^2)$ time.

We now examine the character of the placement problem for river routing when the separation t is given. The n terminals are located on m chunks which are partitioned into two sets that form the top and bottom of the channel. For convenience, we shall number the

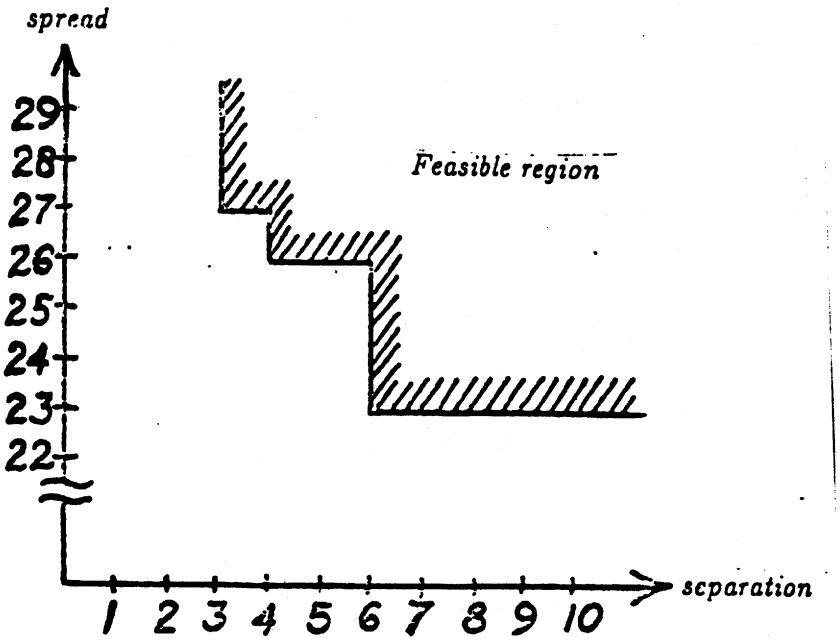
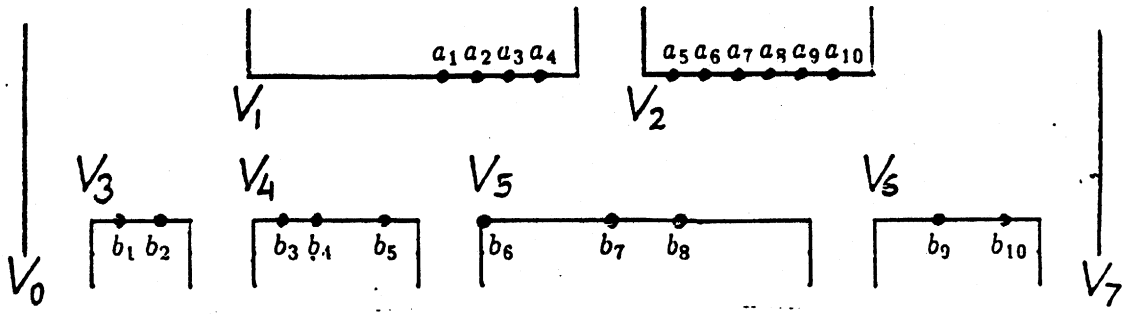


Figure III.7: The curve of minimum spread versus separation for the example of Figure 1.

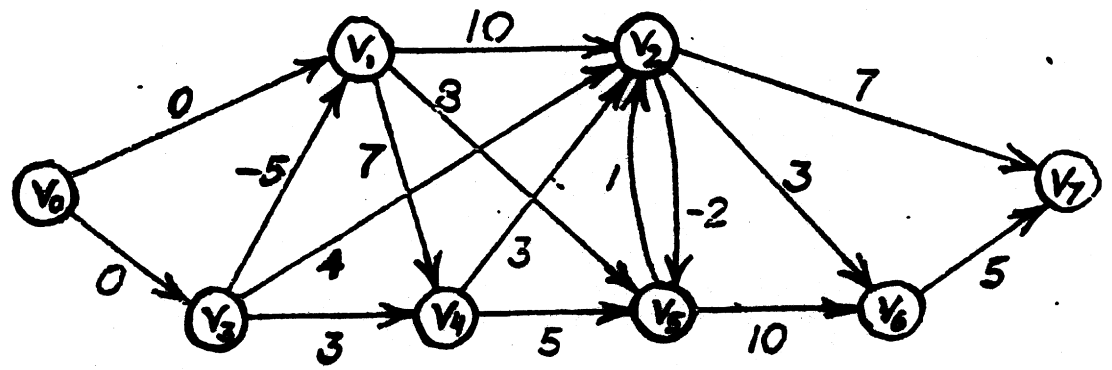
chunks from one to k on the top, and $k+1$ to n on the bottom. The order of chunks on each side of channel is fixed, but they may be moved sideways so long as they do not overlap. For each chunk i, a variable v_i represents the horizontal position of its left edge. Any placement can therefore be specified by an assignment of values to these variables. We also add two variables v_0 and v_{m+1} to the set of variables, which represent the left and right boundaries of the channel. The spread is thus $v_{m+1} - v_0$. Figure 8(a) shows the eight variables for the example from Figure 1.

Since the relative positions of terminals within a chunk is fixed, the wirability constraints of Condition (2) can be reexpressed in terms of the chunks themselves to give placement constraints that any assignment of values to the v_i must satisfy. If terminal a_{i+t} lies on chunk h, and terminal b_i lies on chunk j, the constraint $a_{i+t} - b_i \geq t$ can be rewritten as $v_h - v_j \geq r_{hj}$, where r_{hj} reflects t and the offsets of the terminals from the left edge of their respective chunks. The constraint between two chunks determined in this way will be the maximal constraint induced by pairs of terminals.

Additional constraints arise from the relative positions of chunks on either side of the channel. For each pair of adjacent chunks i and $i+1$, the constraint $v_{i+1} - v_i \geq w_i$ must be added to the set of placement constraints, where w_i is the width of chunk i. Four more constraints are needed which involve the boundary variables v_0 and v_{m+1}. For chunks 1 and $k+1$ which are leftmost on the top and bottom, the constraints $v_1 - v_0 \geq 0$ and



(a) Assignment of variables to chunks and channel boundaries.



(b) The placement graph for separation 3.

Figure III.8: Representing the placement constraints as a graph for the example of Figure 1.

$v_{k+1} - v_0 \geq 0$ enforce that these chunks lie to the right of the left boundary of the channel. For chunks k and m which are rightmost on the top and bottom, the relations $v_{m+1} - v_k \geq w_k$ and $v_{m+1} - v_m \geq w_m$ constrain them to lie to the left of the right boundary, where w_k and w_m are the widths of the chunks.

Figure 8(b) shows a *placement graph* which represents the constraints between chunks for the placement problem of Figure 1 where the separation is 3 tracks. A directed edge with weight δ_{kl} goes from v_k to v_l if there is a constraint of the form $v_l - v_k \geq \delta_{kl}$. For example, the weight of 1 on the *cross edge* going from v_5 to v_2 is the maximal constraint of $a_9 - b_6 \geq 3$ and $a_{10} - b_7 \geq 3$ which yield $v_2 - v_5 \geq -2$ and $v_2 - v_5 \geq 1$ since $a_9 = v_2 + 5$, $a_{10} = v_2 + 6$, $b_6 = v_5$, and $b_7 = v_5 + 4$. The *side edge* from v_4 to v_5 arises from the constraint that chunk 4, which is 5 units long, must not overlap chunk 5.

The goal of the placement problem is to find an assignment of values to the v_i which minimizes the spread $v_{m+1} - v_0$ subject to the set of constraints. This formulation is an instance of linear programming where both the constraints and the objective function involve only differences of variables. Not surprisingly, this problem can be solved more efficiently

than by using general linear programming techniques. In fact, it reduces to a *single-source-longest-paths* problem in the placement graph. The length of a longest path from v_0 to v_{m+1} corresponds to the smallest spread of the channel that complies with all the constraints. The placement of each chunk i relative to the left end of the channel is the longest path from v_0 to v_i. If the placement graph has a cycle of positive weight, then no placement is possible for the given separation.

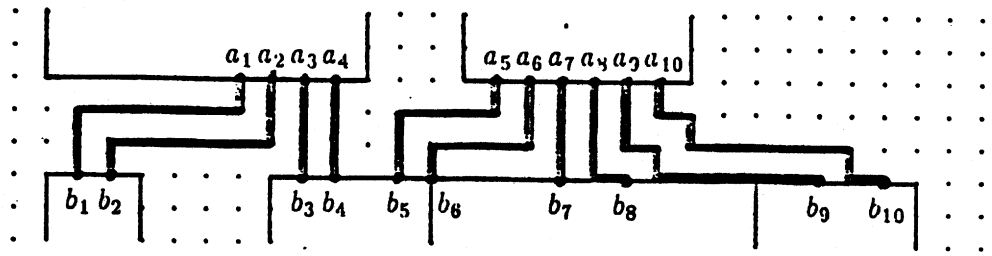
For the placement problem of Figure 1 with a three-track separation, the longest path from v_0 to v_2 in the placement graph (Figure 8) is $v_0 - v_1 - v_4 - v_5 - v_2$ with weight 13 which corresponds to the positioning of chunk 2 in the optimal placement shown in Figure 9(a). Figures 9(b) through 9(d) show optimal solutions to the placement problem of Figure 1 for separations $t = 4$ through $t = 6$. The constraints for $t = 2$ yield a cycle of positive weight in the placement graph, and thus no placement is possible which achieves a separation of only two tracks.

3. A Linear-Time Algorithm for the Fixed-Separation Placement Problem.

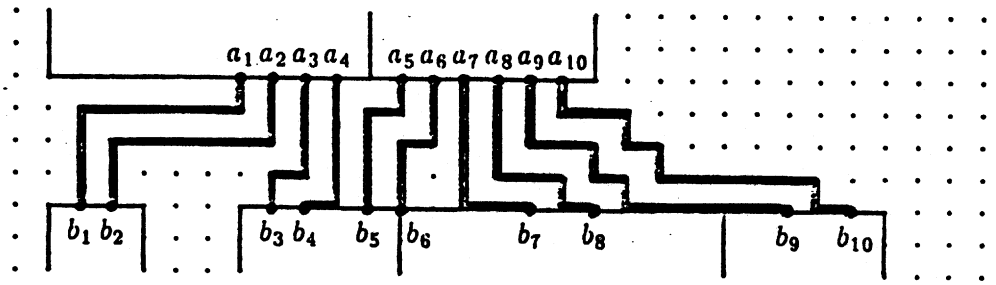
The analysis of Section 2 showed that the optimal placement problem for fixed-separation river routing was reducible to the single-source-longest-paths problem on a placement graph. For a general graph $G = (V, E)$ this problem can be solved in time $O(|V| \cdot |E|)$ by a *Bellman-Ford algorithm* [La76]. Better performance is possible, however, due to the special structure of placement graphs. This section reviews the Bellman-Ford algorithm, and shows how it can be adapted to give an $O(m)$-time algorithm for the longest-paths problem on a placement graph, where m is the number of chunks. Since the placement constraints can be generated in $O(n)$ time, where n is the number of terminal pairs, this algorithm leads to an optimal linear-time algorithm for the fixed-separation placement problem.

The linear-time algorithm is a refinement of the standard Bellman-Ford algorithm which for each vertex v_i where $i = 1, \ldots, m+1$, iteratively updates the length $\lambda(v_i)$ of a tentative longest path from v_0 to v_i. The algorithm initializes $\lambda(v_0)$ to zero, and all other $\lambda(v_i)$ to $-\infty$; then it sequences through a list $\mathcal{E}$ of edges, and for each edge (v_i, v_j) with weight δ_{ij} updates $\lambda(v_j)$ by

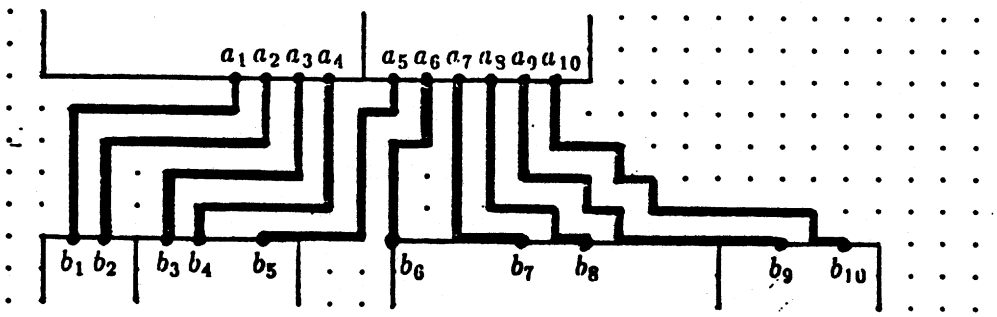
$$\lambda(v_j) \leftarrow \max(\lambda(v_j), \delta_{ij} + \lambda(v_i)).$$



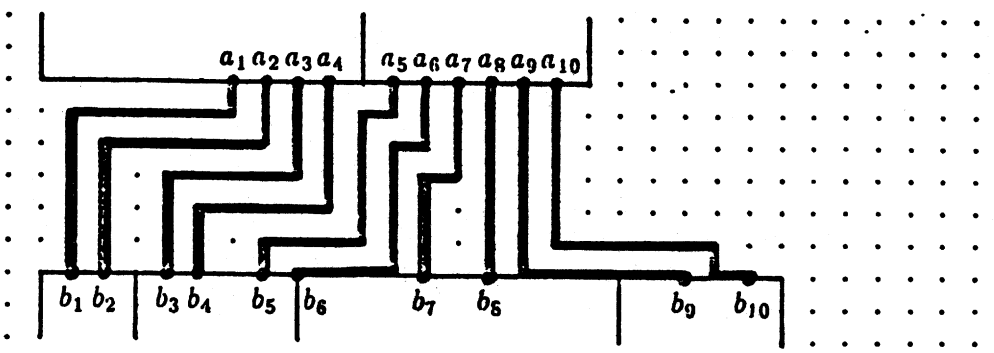
(a) Separation 3, spread 27.



(b) Separation 4, spread 26.



(c) Separation 5, spread 26.



(d) Separation 6, spread 23.

Figure III.9: Optimal placements and routings for the problem of Figure 1 with separations ranging from $t = 3$ to $t = 6$.

The list $\mathcal{E}$ of edges is the key to the correctness of the algorithm. *The length of a longest path from the source v_0 to a vertex v_j converges to the correct value if the edges of the path form a subsequence of the list $\mathcal{E}$.* (This can be proved by adapting the analysis of [Ye70].) In the normal algorithm for a general graph $G = (V, E)$, the list $\mathcal{E}$ is $|V| - 1$ repetitions of an arbitrary ordering of the edges in E, which ensures that every vertex-disjoint path in G beginning with v_0 is a subsequence of $\mathcal{E}$. If there are no cycles of positive weight in the graph G, then from v_0 to each other vertex in G, there is a longest path that is vertex-disjoint; hence the algorithm is guaranteed to succeed. The condition of positive-weight cycles can be tested at the end of the algorithm either by checking whether all constraints are satisfied or by simply running the algorithm through the edges in E one additional time and testing whether the values of any $\lambda(v_i)$ change.

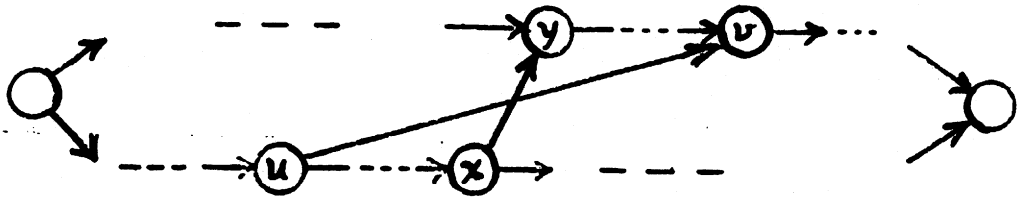
The list $\mathcal{E}$ is also the key to the performance of a Bellman-Ford algorithm. For the general algorithm on an arbitrary graph $G = (V, E)$, the length of the list is $(|V| - 1) \cdot |E|$, and thus the algorithm runs in $O(|V| \cdot |E|)$ time. For a placement graph it is not difficult to show that both $|V|$ and $|E|$ are $O(m)$, and thus the longest-paths problem can be solved in $O(m^2)$ time by the general algorithm. But a linear-time algorithm can be found by exploiting the special structure of a placement graph to construct a list $\mathcal{E}$ of length $O(m)$ that guarantees the correctness of the Bellman-Ford algorithm. We now look at the structure of placement graphs more closely.

The vertices of a placement graph $G = (V, E)$ corresponding to the chunks on the top of the channel have a natural linear order imposed by the left-to-right order of the chunks. We define the partial order $\prec$ as the union of this linear order with the similar linear order of bottom vertices. Thus $u \prec v$ for vertices u and v if their chunks lie on the same side of the channel and the chunk that corresponds to u lies to the left of the one which corresponds to v. The left-boundary vertex v_0 precedes all other vertices, and all vertices precede the right-boundary vertex v_{m+1}. The partial order $\preceq$ is the natural extension to $\prec$ that includes equality.

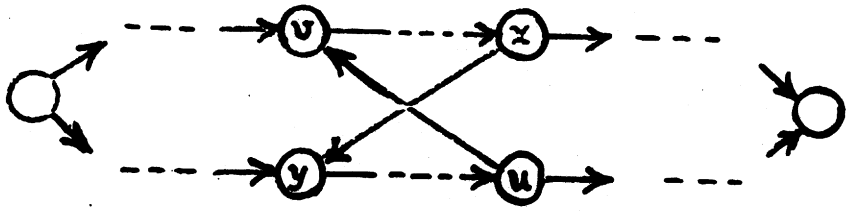
The next lemma describes some of the structural properties of placement graphs. Figure 10 illustrates the impossible situations described in Properties (*i*) and (*ii*) and shows the only kind of simple cycle that can occur in a placement graph together with the two consecutive cross edges that satisfy Property (*iii*).

Lemma III.1. *Any placement graph $G = (V, E)$ has the following properties:*

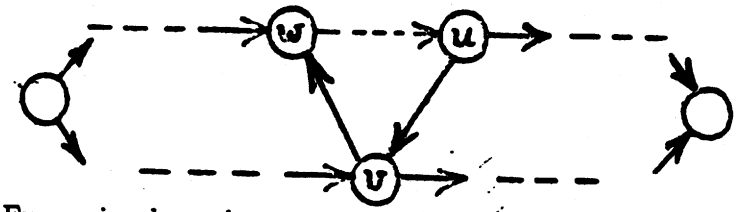
- (*i*) *There do not exist cross edges (u, v) and (x, y) such that $u \prec x$ and $y \prec v$.*
- (*ii*) *There do not exist cross edges (u, v) and (x, y) such that $v \prec x$ and $y \prec u$.*
- (*iii*) *All cycles have two consecutive cross edges (u, v) and (v, w) such that $w \preceq u$.*



(a) The situation forbidden by Property (*i*).



(b) The situation forbidden by Property (*ii*).



(c) Every simple cycle contains at most one vertex from the top or at most one vertex from the bottom. The edges incident on the vertex are a consequence of Property (*iii*).

Figure III.10: The properties of the placement graph enumerated in Lemma 1.

Proof. Properties (*i*) and (*ii*) can be proved by considering which of the terminal constraints from Condition (2) induce the edges in the placement graph. For each of these cases, suppose the edge (u, v) was caused by the terminals i in u and $i + t$ in v, and the edge (x, y) came from the terminals j in x and $j + t$ in y. For Property (*i*) we have $u \prec x$ and $y \prec v$, and thus $i < j$ and $j + t < i + t$. Canceling t from this latter inequality obtains the contradiction. The assumption to be proved impossible in (*ii*) is that $v \prec x$ and $y \prec u$, which implies $i + t < j$ and $j + t < i$. Since t is nonnegative, we gain a contradiction.

To prove Property (*iii*), we need only consider simple (vertex-disjoint) cycles. Since no cycle can consist solely of side edges, every simple cycle must have a cross edge (u, v) going from bottom to top. In order to complete the cycle, there must be a top-to-bottom edge (w, x) such that $v \preceq w$ and $x \preceq u$. If $v = w$ or $x = u$, then the pair of edges satisfies Property (*iii*). But if $v \neq w$ and $x \neq u$, then the pair of edges violates Property (*ii*). ∎

Each edge in the placement graph is either a top edge, a top-bottom edge, a bottom-top edge, or a bottom edge. For each of these four sets of edges, there is a natural linear order of edges based on $\preceq$, where (u, v) precedes (x, y) for two edges in the same set if $u \preceq x$ and

$v \preceq y$. Property (*ii*) guarantees that the linear order holds for two cross edges in the same set. Let TT, TB, BT, and BB be the four lists of edges according to the natural linear order, and include the two edges out of v_0 and the two edges into v_{m+1} in either TT or BB as appropriate.

The list $\mathcal{E}$ used by the Bellman-Ford algorithm is constructed by a merge of the four lists which we call MERGE. At each step of MERGE, a tournament is played among the first elements of each list. If (u, v) and (v, w) are the first elements of two lists, then (u, v) beats (v, w) if $w \not\preceq u$. Since there may be more than one edge beaten by none of the other three, ties are broken arbitrarily. The winner is appended to $\mathcal{E}$ and removed from the head of its list. The tournament is then repeated until no edges remain in any of the four lists. The performance of the tournament can be improved by recognizing that only six of the twelve possible comparisons of edges need be tried, and that $w \not\preceq u$ is guaranteed for all but two. Figure 11 shows a possible ordering of edges in $\mathcal{E}$ for the placement graph in Figure 8.

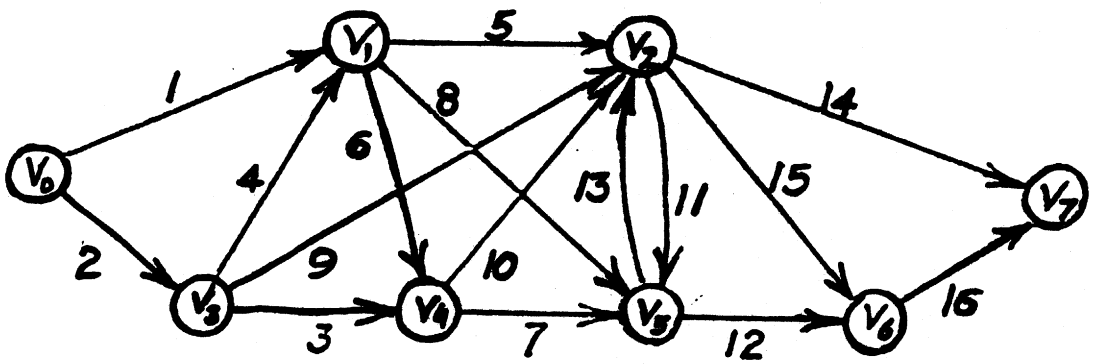


Figure III.11: A possible ordering of edges in $\mathcal{E}$ for the placement graph in Figure 8.

In order for MERGE to be well-defined, the tournament must always produce a winner, which is a consequence of the next lemma.

Lemma III.2. *The list $\mathcal{E}$ produced by* MERGE *is a topological sort of the edges of E according to the relation R where $(u, v)R(v, w)$ if $w \not\preceq u$.*

Proof. First, we show that the relation R is acyclic so that the edges can indeed be topologically sorted. By definition of R, a cycle in R induces a cycle in the placement graph. According to Property (*iii*), the cycle must have two consecutive cross edges (u, v) and (v, w) such that $w \preceq u$. But since $(u, v)R(v, w)$, we also have that $w \not\preceq u$, which is a contradiction.

The proof that MERGE topologically sorts the edges of E according to R makes use of the fact that if a vertex v is the tail of an arbitrary edge in any one of the four lists TT, TB, BT or BB, then for every $u \preceq v$ there is an edge in the same list emanating from u. Suppose that MERGE does not topologically sort the edges of E according to R. Then there is a first

edge (u, v) in $\mathcal{E}$ such that there exists an edge (v, w) earlier in $\mathcal{E}$ and $(u, v)R(v, w)$. Consider the edge (x, y) in the same list as (u, v) that competed with (v, w) when (v, w) was the winner of the tournament. For each of the possible combinations of lists for (u, v) and (v, w), it can always be deduced that there is an edge emanating from y such that which makes (x, y) an earlier violator of the topological sort than (u, v). ∎

Since each edge of E is included exactly once in the list $\mathcal{E}$ created by MERGE, the Bellman-Ford algorithm applied to $\mathcal{E}$ has a running time linear in the number of chunks. The correct values for longest paths are produced by the algorithm if for every vertex v, there is a subsequence of $\mathcal{E}$ that realizes a longest path from v_0 to v, under the assumption that there are no positive-weight cycles in the placement graph. Since for every longest path, there is a vertex-disjoint longest path, the following theorem proves the correctness of this linear-time Bellman-Ford algorithm.

Theorem III.3. *Let G be a placement graph with left-boundary vertex v_0. Then every vertex-disjoint path beginning with v_0 is a subsequence of the list $\mathcal{E}$ created by the procedure* MERGE.

Proof. We need only show that every pair of consecutive edges in a vertex-disjoint path from v_0 satisfies R because then Lemma 2 guarantees that the path is a subsequence of $\mathcal{E}$. Suppose (u, v) and (v, w) are two consecutive edges on a vertex-disjoint path from v_0 which violate R, that is, $w \preceq u$. If either (u, v) or (v, w) is a side edge, the pair must satisfy R, and thus both must be cross edges with the vertices u and w on the same side. Since if $w = u$, the path is not vertex-disjoint, we need only show that $w \prec u$ is impossible.

Assume, therefore, that $w \prec u$, and consider the initial portion of the path from v_0 to u. Since $v_0 \prec v$ and $v_0 \prec w$, there must be an edge (x, y) on the path which goes from the set of vertices to the left of (v, w) to the right of (v, w) in order to get to u. But then either Property (i) or Property (ii) is violated depending on whether $x \prec v$ and $w \prec y$, or $y \prec v$ and $x \prec w$. ∎

4. The Placement Problem for Multiple Parallel Channels

Multiple, parallel horizontal channels (Figure 12) are easily handled within the same graph-theoretic framework as long as the width of each channel is given. Every row of chunks is represented by a chain of vertices from a common left boundary to a common right boundary. The wiring conditions in the channels are represented by edges linking adjacent chains. The optimal placement is achieved by solving the longest paths problem on this graph. The standard Bellman-Ford algorithm runs in time $O(n + m^2)$ since the number of edges in

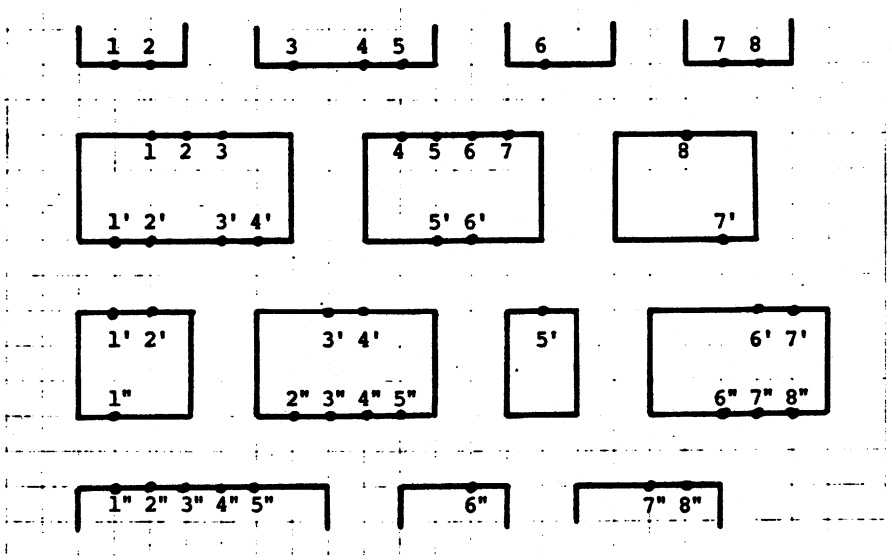


Figure III.12: Multiple horizontal channels.

the graph is $O(m)$. We do not know whether improvements of the kind used in Section 3 for the single channel problem can be obtained for the multiple channels case.

The problem is much harder, however, when the input is specified so that only the *sum* of the channel separations is given. Then the problem is to allocate the channel widths so as to minimize the spread subject to the additional constraint on the channels' total width. This problem is shown here to be NP-complete. The NP-completeness result was obtained jointly with Michael Sipser.

The problem can be stated formally as follows:

[PDP] **PLACEMENT FOR A DATA PATH**

INSTANCE: A set of modules arranged in $k + 1$ rows, with all modules in the same row having the same height. Each module (except those on the first and last rows) faces two channels — one at its top, the other at its bottom, with terminals on those sides. The top row has terminals only on the bottom of modules, the last — only on their top. All k channels are river routable (so the *order* of modules in each row is fixed). The dimensions of the total routing area are also given: the sum of the widths of all channels, t, and a spread, s.

QUESTION: Is there a placement for the modules (in straight rows) such that all channels are routable, the sum of their widths does not exceed the prespecified total width t, and the total extent of the layout in the horizontal dimension (*i.e.* the distance from the leftmost edge of a module to the rightmost edge) does not exceed s?

The transformation* is from Satisfiability of Boolean Expressions [GaJo79, pp.260–261]. The set of binary connectives we use is $\{\rightarrow, \neg\}$. The given formula $\mathcal{F}$ has p variables and q nodes in its parse tree.

The data path placement problem that we construct can be described by a framework into which the modules are being put. Each variable in $\mathcal{F}$ is represented by a *column* consisting of one module in each row. Columns are separated by *foundations* of the framework that ensure that no lateral movement of variable-modules is allowed beyond the extent of the column. Each column *carries* a signal by terminals that keep all modules in a column vertically aligned. A truth assignment of a variable corresponds to whether the modules in the column are justified to the left (true) or right (false) margin of the column. We find the parse tree with the smallest number of intermediate terms, and add one extra column for each such term.

We ensure the vertical alignment of variable-modules in a column and of the parts of the foundation by connecting them with a large number of nets that are tightly packed along the modules' sides at the same offset from the left hand corner. The exact number will be calculated later. Recall that the horizontal alignment of modules in a row is required by the problem specification.

The *operators* ($\rightarrow$ and $\neg$) are realized by the channels, two channels per operation. The interaction between a variable and its surrounding foundation at the intersection with the channel is the key for realizing an operation. We demonstrate this by exposing the construction for the implication, illustrated in Figure 13 which shows the construction for $x_1 \rightarrow x_2$. x_1 is realized by the column on the left, and x_2 — by the column on the right. Across the top channel of the operation x_1 is connected to a part of the foundation that is to its right. x_2 is connected to the foundation on its left across the bottom channel. If the modules for x_1 are on the left, the separation of the top channel has to be at least 1. If we want the combined separation of the two channels to be at most 1, the modules representing x_2 have to be left justified in their column. If, on the other hand, the modules for x_1 are on the right (meaning x_1 is false), x_2 can be assigned any value. A similar construction realizes the negation operation.

Thus we set the spread to be the width of the assembly, and the separation is half the number of channels. The number of terminals that connect across a channel between two modules of the same variable or two blocks of the foundation is one more than the number of channels. This way any additive tradeoffs between channels are being ruled out, and the formula $\mathcal{F}$ can be satisfied if and only if the data path can be laid out within the specified dimensions.

---

* We use the terminology of [GaJo79], therefore "transformation" and not "reduction".

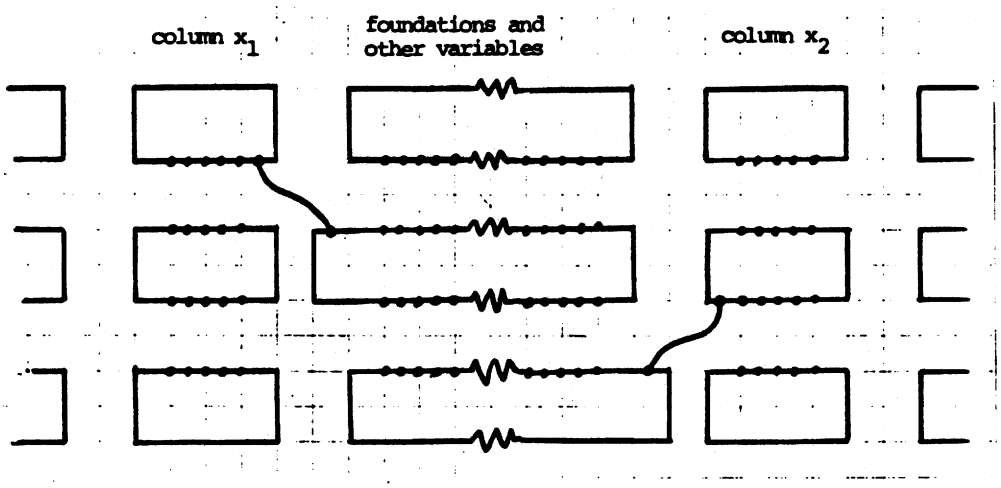


Figure III.13: The realization of $x_1 \rightarrow x_2$ by two channels in the data path placement problem.

5. Nonrectilinear and Multilayer Wiring Models

For a single channel, the reduction from the fixed-separation placement problem in the square-grid model to the single-source-longest-paths problem is possible because the wirability constraints can all be written in the form $v_i - v_j \geq \delta_{ij}$. Thus for any wiring model where wiring constraints can be written in this form, the reduction will succeed. Also, it should be observed that in general, the performance of the single-source-longest-path algorithm will not be linear, but will be a function of the number of constraints times the number of variables. This section reviews other models and gives the necessary and sufficient wirability constraints for each. Some of these models are discussed in [Bar81], [DKSSU81], [SieDo81], and [To80].

1. *One-layer, gridless rectilinear.* Wires in this model must run horizontally or vertically, and although they need not run on grid points, no two wires can come within one unit of each other. The wirability constraints for this model are the same as for the square grid model:

$$a_{i+t} - b_i \geq t \quad \text{and} \quad b_{i+t} - a_i \geq t$$

for $1 \leq i \leq n - t$. As with the square-grid model, the fixed-separation placement algorithm for this model can be made to run in linear time.

2. *One-layer, gridless, rectilinear and forty-five degree.* This model is the same as the gridless rectilinear, but in addition wires can run which have slope ± 1. The constraints in this case are

$$a_{i+r} - b_i \geq r\sqrt{2} - t \quad \text{and} \quad b_{i+r} - a_i \geq r\sqrt{2} - t$$

for $t/\sqrt{2} \leq r \leq t$ and $1 \leq i \leq n - r$. The placement algorithm for this model runs in $O(\min(tm^2 + tn, m^3 + tn, m^3 + n^2))$ time.

3. *One-layer, gridless.* Wires can travel any direction. The constraints are

$$a_{i+r} - b_i \geq \sqrt{r^2 - t^2} \quad \text{and} \quad b_{i+r} - a_i \geq \sqrt{r^2 - t^2}$$

for $t \leq r \leq n$ and $1 \leq i \leq n - r$. The placement algorithm runs in $O(m^3 + n^2)$ time.

4. *Multilayer models.* All the models presented until now have been one-layer models. It is natural to generalize to l-layer models in which wires may travel on different layers. Remarkably, optimal routability can always be achieved with no contact cuts [Bar81], that is, a wire need never switch layers. The necessary and sufficient conditions for these multilayer models are a natural extension of the one-layer conditions. For example, in the one-layer, gridless, rectilinear model the conditions are modified for l layers to be

$$a_{i+lt} - b_i \geq t \quad \text{and} \quad b_{i+lt} - a_i \geq t$$

for $1 \leq i \leq n - lt$.

There are some wiring models, however, where upper and lower bounds for wirability do not meet. For these models a constraint graph which represents upper bounds will give the best possible placement for those bounds. A graph representing lower bounds will give lower bounds on the best possible placement. Together, bounds can be established for some of these models, and heuristic algorithms invoked to attempt routing within the feasible range of optimality.

6. Extensions and Conclusions

A variety of related placement problems can be solved by the method described in this chapter. Some entail extensions to the problem specifications, others employ different wiring models. In this section we shall mention a few extensions we can handle and suggest further research on more complicated problems. Some of these problems are explored in other chapters of this thesis, and appropriate references are given.

- *Nonriver routing.* The placement algorithm gives optimal placements for river routing, but there are other routing configurations for which it works optimally as well. One example is the two-layer, any-to-any routing problem where two sets of terminals must be connected across a channel, but they may be connected in any order. More on this can be found in Chapter VI.

- *Range-terminals.* In some routing situations terminals occupy not a single point, but rather a contiguous region along the edge of the channel. For example, the terminal might be a wire that runs along the edge of the chunk, and connection can be made to the wire

anywhere. The additional flexibility of viewing a terminal terminal as a contiguous range of points can be exploited by both the greedy routing algorithm and the placement algorithm in any of the river-routing models we have discussed.

Each range-terminal is specified by an interval $[a_i^L, a_i^R]$ or $[b_i^L, b_i^R]$. The greedy routing algorithm operates as before with minor changes. If the range-terminals overlap, the wire is routed straight across. Otherwise, assume without loss of generality that $a_i^R < b_i^L$, and use the standard greedy algorithm to route a wire from a_i^R to b_i^L.

The wirability conditions for placement are accordingly adjusted. In the rectilinear case, for example, the condition $a_{i+t} - b_i \geq t$ is rewritten as $a_{i+t}^R - b_i^L \geq t$ and condition $b_{i+t} - a_i \geq t$ becomes $b_{i+t}^R - a_i^L \geq t$. The transformation to chunk variables is as before and the placement algorithm is unchanged.

- *Variable-width wires.* In some applications the wires that must be routed do not all have the same width. Our scheme can be generalized to deal with this situation as long as each wire has uniform width. Both routability and placement for a fixed-separation problem can be determined in linear time by computing an experimental cumulative distribution function (ECDF) of the wire widths.

The wirability conditions have to be changed and all coordinates are real numbers, not grid points as before*. The ECDF is obtained by drawing parallel wire segments having the widths of the terminals $a_1, \ldots, a_n$ at minimum spacing between them, as shown in Figure 14. We denote the x-coordinate of the left end of net i's representative by l_i, and align the diagram by $l_1 = 0$.

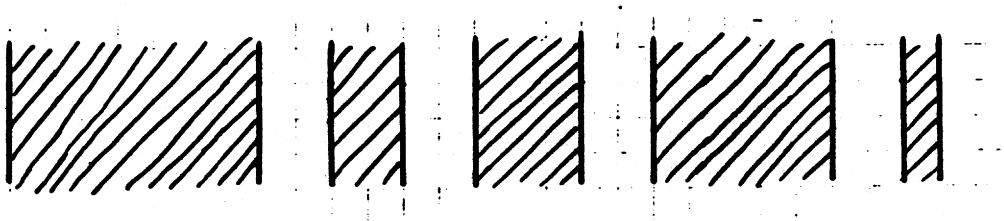


Figure III.14: The experimental cumulative distribution function for five wires of widths 7,2,3,5,1. The spacing required between wires is 2 units, thus $l_2 = 9$, $l_3 = 13$, etc.

For a fixed separation t, we associate with each net i the variable $x_i = l_i + t$. To test routability we draw forty-five degree rays from the left corner of each terminal. For terminal a_i, we find the terminal b_j closest on the left to the intersection point on the bottom, and set d to the distance between (the left corner of) b_j and the intersection point. We require that $d \geq 0$. Then we set $x_j = l_j + d$, and the routability condition is that $x_i \leq x_j$ for all nets i

* A finer grid, such as the λ-grid used in [MeaCo80] could be used instead of real numbers.

and their corresponding x_j's. The test has to be repeated by applying it to all b_i's as well (in place of the a_i's above).

- *Minimum jogging.* The number of jogs (or turns) of wires produced by the greedy algorithm may be excessive. In some cases, $\Omega(n^2)$ jogs will be produced when $O(n)$ jogs suffice to route the same channel. An adaptation of the routability constraints developed in Section 1 can be used to absolutely minimize the total number of jogs in the channel; this is described in Section V.4.

- *River-routing in a polygon.* Instead of constraining terminals to lie on two parallel lines, we allow them to reside anywhere along the boundary of a simple polygon. The planarity of the interconnect as well as the wirability within the polygon's area can be tested in time $O(n + p)$, where p is the number of corners in the polygon. A routing can be produced in time $O(n^2 + pn)$ using an extension of the greedy algorithm. These results are described in Chapter IV.

- *Two-dimensional river routing.* The two-dimensional river-routing problem is illustrated in Figure 15. In the figure, a line between two chunks indicates that wires must be river-routed between them. Unfortunately, in order to optimally solve this general problem, it appears that the constraints indicated by the lines must be convex in both dimensions, not just in one as is the case for the wiring models considered here. When the constraints are convex, however, convex programming can be used to optimize a cost function such as the area of the bounding box of the layout. One model which gives convex constraints for the general two-dimensional problem is the one in which all wires must be routed as straight line segments between terminals such that no minimum spacing rules are violated. This model is not particularly interesting from a practical standpoint, however. Heuristics for solving the related two-dimensional *compaction* problem by repeatedly compacting in one dimension and then the other can be found in [Hs79].

A different perspective on the two-dimensional river routing (and related placement issues other then stretching) is explored in the next chapter.

A major deficiency of many placement programs is that they lack knowledge about the wirability of the routing problems that they set up. We have shown for river routing that wirability conditions can be translated directly into placement constraints without the overhead of actually wiring the channel. For rectilinear river routing, the running time of the greedy wiring algorithm is $O(n^2)$, and any cost function for placement that is monotonic in spread and separation can be optimized in $O(n^2)$ time without the overhead of routing. Studying wirability in the general case may lead to the development of heuristics for wirability

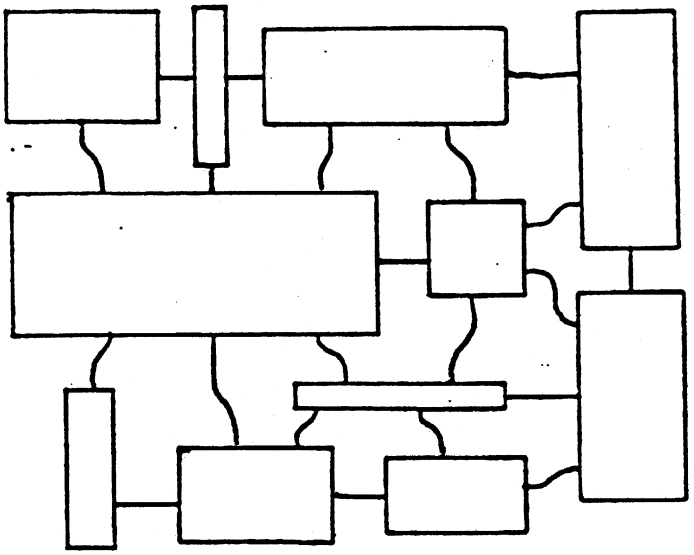


Figure III.15: A two-dimensional extension to the river-routing problem. A solid line between two modules indicates routing occurs between them.

that do not involve routing. A program that uses this heuristic knowledge should be able to outperform the iterative place-route, place-route programs that dominate today.

Chapter IV

Generalized River Routing

The problem of river routing across a channel, as studied in Chapter III, is only a special case of more general routing configurations. Both its methodological and combinatorial characteristics can be extended in useful ways which will be explored in this chapter. The two characteristics that we generalize here are planarity and grouping. *Planarity* means that the connections are realizable in one layer, *i.e.* the interconnection pattern of the nets is planar. *Grouping*, which is a restriction of the planarity characteristic but deserves attention in its own right, means that the connections are made *in order*, that is to say that the routing of net $i+1$ is adjacent, conceptually and preferably physically, to the routing of net i.

This chapter investigates placement and routing problems within the same framework. First we provide a graph theoretical model that accomodates the interconnect specifications in a succinct manner, allowing us to find a placement that enables a planar routing pattern in linear time. Second we study problems of detailed routing, namely whether wires fit in the area allotted by a specific placement. Detailed planar routing of two-point nets for an entire chip (with rectangular modules) is shown to be NP-complete, whereas a polynomial time algorithm is given for detailed routing for a simple polygon (without holes). Routability testing is shown to be easier than the actual generation of the wires.

1. Introduction

The issue of planarity is not new. Much of the attention devoted to the subject in the context of the design of electrical circuits dates back to the PCB era, when layer changes were expensive

and degraded the quality of products. But even in the current MOS technologies single layer realizations have non-negligible advantages, because one layer (metal) is highly preferable to the others.

To discuss the grouping issue, we need to define cables and sequences. A maximal set of nets that connects one sequence of terminals on a side of a module *in order* to another sequence on another module, is called a *cable*. A *sequence* of terminals encapsulates the notions of contiguity and of a geometric order between the terminals. For sake of presentation, we confine ourselves to sequences that lie along *one* side of *one* module. In other words, a sequence cannot be split between modules and is lined up along a straight piece of a module's boundary. Both these restrictions are not essential, since more general notions of sequences can be constructed using this basic type without impairing the validity of the technical results presented in this chapter.

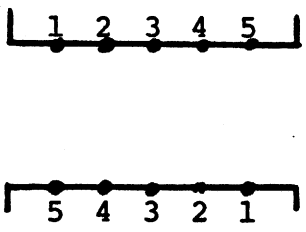
Cables occur frequently in many designs, most notably in the design of microprocessors where fields of data have to be transmitted from one place to another. The recognition of routing patterns such as cables constitutes an abstraction mechanism by which some of the low level details are suppressed in favor of a clear perception of the architectural issues. Such an abstraction is important not only from a methodological point of view, but also useful in reducing the complexity of solutions to several problems. In addition, it is easier to control the circuit level performance of a parallel connection if the realization of all signals involved is similar, *i.e.* the wires are known to have similar characteristics in terms of layering, length, and jogging.

In Section III.1 we saw that the problem of routing a cable across a channel can be solved efficiently and optimally. One salient assumption in that setting was that the orderings of terminals on both sides conformed in such a way that a planar realization was possible (provided there was enough room). In the general case, we must decide whether this is the case to start with. This test has two parts. First, "Is there a passage between the locations of the involved sides?", and second, "Are the sides oriented properly with respect to each other?" Figure 1(a) demonstrates how the orientability issue comes up even in the channel case. When the sides are further apart on the chip this problem may look harder (see Figure 1(b) and (c)), but in fact it is not. The problem can be formalized as follows:

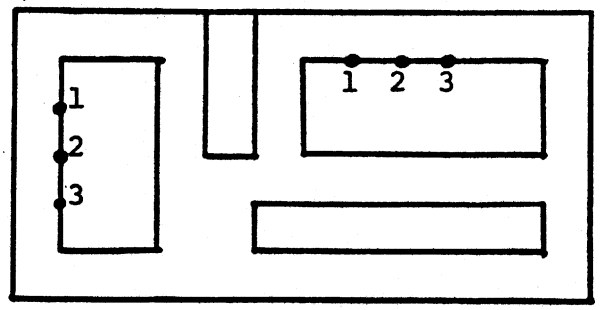
[CR] **CABLE ROUTABILITY**

INSTANCE: A placement of a set of rectangular modules within a bounding box, and a cable connecting two sequences on (not necessarily disjoint) sides of modules. The n terminals of each sequence are numbered in order from 1 to n.

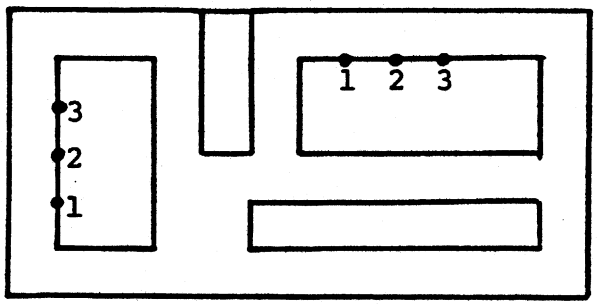
QUESTION: Is there a planar (one-layer) realization for this cable?



(a) A cable crosses a channel the wrong way.



(b) properly oriented cable



(c) improperly oriented cable

Figure IV.1: Two-dimensional extensions to the river-routing problem.

In a more general setting, however, we may have the freedom of orienting the modules in such a way that will enable routing in the plane. Recall (from Section II.1) that the operations that may be performed on a module before it is placed are (rectilinear) rotation and reflection. After a module is placed we may modify its position by *translation*, which is the ability to move the module in the plane without changing its orientation. Reflection is essential in ensuring the feasibility of planar embeddings from a topological point of view, whereas translation and rotation may be required from a geometric viewpoint—to make room for wires (as we have seen, in part, in Chapter III). All three operations are realizable at the mask level for integrated circuits without affecting any of the physical aspects of the design. Reflection, however, may not be as easy to realize in a PCB design, which is a major concern since this very operation is the one that is most important for the possibility of planar embedding.

The question to be asked, then, is how can these three operations (or a subset thereof, excluding rotation) be used in generating a placement (or modifying an existing one) in which

all cables are routable in the plane. First we state the problems concerning the topological issue alone. For the feasibility of planar interconnect we distinguish between the case in which the placement is given and the case where we are free to determine it.

[PR] PLANAR ROUTABILITY

INSTANCE: A placement of a set of rectangular modules within a bounding box, numbered terminals on modules' boundaries.

QUESTION: Is there a planar embedding of the interconnect wires for this given placement?

[PO] PLACEMENT ORIENTATION

INSTANCE: A set of (non-oriented, flippable) rectangular modules, numbered terminals on each modules' boundary.

QUESTION: Is there an embedding of the modules on the plane such that the routing is also feasible in the plane?

Once modules have been oriented and placed in such a way that a planar embedding is topologically feasible, there still remains the question of fitting the actual wires in the given area. Naturally, the placement found to be consistent with a one layer realization induces a set of possible routing paths for the nets. However, such a set may not be uniquely determined by the topology suggested by the placement, but even if the paths are unique, the actual placement may impose some further constraints on the actual routing. Thus, the problem of detailed routing arises at two different levels. First, the more general problem is that of finding a routing for a given placement, where each wire can be routed wherever it fits:

[DR] DETAILED ROUTING

INSTANCE: A placement of a set of rectangular modules within a bounding box, numbered terminals on modules' boundaries.

QUESTION: Is there a one-layer detailed routing for this configuration?

The second detailed routing problem starts off with more information about the routing plan. For every net, we are given the *homotopy* of its intended wire relative to the placed modules. That is to say, we know how the path taken by the wire is related to the positions of the modules — how it goes "between" pairs, whether it goes to the "right" of or "above" a module etc. Figure 2 exemplifies the notion of a homotopy. A given homotopy of a wire is sometimes called a *rough routing* of the net (as opposed to detailed routing which entails specifying the exact path). Also,

we say that we are given a rough routing for a problem if all its nets are roughly routed. This notion corresponds to that of *global routing* which is commonly used in the literature in the context of the general (nonplanar) routing problem (see, for example, [SouRo81], [Riv82]). The second detailed routing problem, then, can be formulated as follows:

[DRH] **DETAILED ROUTING GIVEN A HOMOTOPY**

INSTANCE: A placement of a set of rectangular modules within a bounding box, numbered terminals on modules' boundaries, homotopy (rough routing) for each net.

QUESTION: Is there a one-layer detailed routing for this configuration that conforms with the given homotopy?

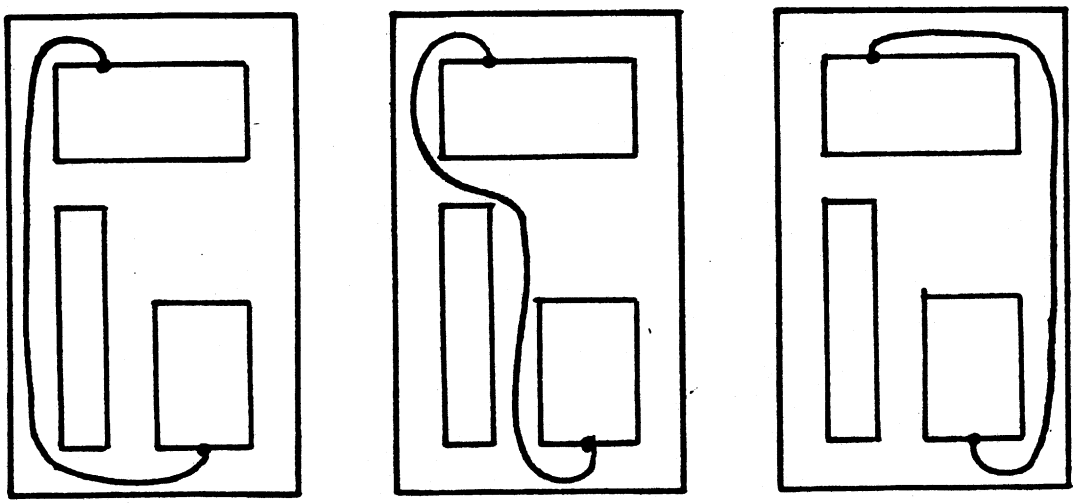


Figure IV.2: Different homotopies for the same wire.

When specifying a rough routing, the cable abstraction comes in handy. Cables provide a succinct manner by which the homotopy of a set of "parallel" wires can be specified, and moreover, many of the details concerning the relative position between wires in the same cable are rendered unnecessary.

A natural restriction of the planar routing problem is the situation in which there is no difference between DR and DRH. This is the case in which routing is being performed in a simply connected polygon, where there are no "holes" to route around and the rough routing for each net is trivially unique (topologically). Figure 3 shows such a case in which planarity is guaranteed. The formulation of the decision problem is given as follows:

[DRSRP]**DETAILED ROUTING IN A SIMPLE RECTILINEAR POLYGON**

INSTANCE: A simple rectilinear polygon with terminals on its boundary.

QUESTION: Is there a one-layer detailed routing for this configuration?

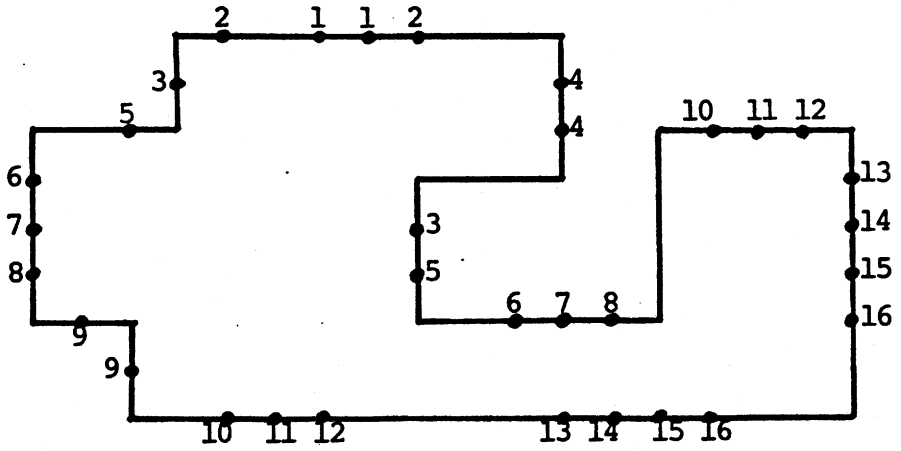


Figure IV.3: The specification of a planar routing problem in a simple polygon.

The rest of this chapter is devoted to investigating and classifying the complexity of the problems stated so far. We show that DR is NP-complete, and present polynomial time algorithms for CR, PR, PO, and DRSRP. The classification of DRH remains open.

For those problems that are solvable in polynomial time, there may be a qualitative difference between the time it takes to decide the problem and the time it takes to produce an actual solution (in the affirmative case). We have experienced this phenomenon already in Chapter III when we realized that testing for river routability across a channel can be done faster than producing the actual layout. Such distinctions concerning the problems of this chapter will be pointed out as we go along.

2. Placement

We start off here by studying the problem of cable routability (CR) in Subsection 1. Solving it will provide us with the proper insight required in order to study the placement problems, PR and PO. We shall be able to use the solution to CR in formulating the orientability constraints that need to be satisfied by any feasible placement. These constraints, when applied to a given set of modules (with a given orientation), may force some of the modules to be reflected. Such requirements can be straightforwardly accomodated in a graph theoretical framework to be developed in Subsection 2. The case in which modules are not allowed to be reflected can be solved within the same framework, but some extra work is necessary to ensure that the original

orientation is preserved. This is done in Subsection 3 by modifying the planarity testing algorithm due to Hopcroft and Tarjan ([HoTa74], [Ev79, Chapter 8]).

2.1. Cable Routability

Two problems need to be solved in order to resolve the cable routability (CR) problem: *reachability* and *orientability*. Reachability is the problem of deciding whether the side on which one end of the cable lies on can be connected to the other side by some path travelling through the routing area. This can be solved easily by finding the connected parts of the (not necessarily simple) polygon that comprises the routing area, and testing whether the two sides belong to the same component. The overall complexity of the reachability test is $O(m\alpha(m)$*. This can be achieved by partitioning the routing area into $O(m)$ stripes and then merging them using a union–find algorithm.

The orientability problem asks whether the sequences at both ends of the cable are oriented in a way consistent with a planar realization. This has been exemplified in Figure 1 above.

First, we observe that there is a qualitative difference between cables comprised of two nets on one hand, and cables with three or more wires on the other hand. Two terminals can usually be permuted to obtain the two possible sequences on any side of a module and still be realizable in the plane, but three terminals or more cannot. The reason for this is that whereas two nets can be realized both as one single cable and as two separate cables consisting of one wire each, and still fit in the plane, with three nets any split that maintains the sequence property at both ends of the cable will cause a violation of planarity as a consequence of Kuratowski's Theorem (see, for example, [Ev79, Chapter 7]). Figure 4 shows how a two-net cable can be realized in the plane regardless of the relative orientations of the sequences at its ends — once as defined (a) and once by splitting it into two cables (b). A three-net cable, however, must be oriented properly (parts (c) and (d) of Figure 4).

* α is the inverse of Ackermann's function, and can be regarded as a constant for all practical purposes.

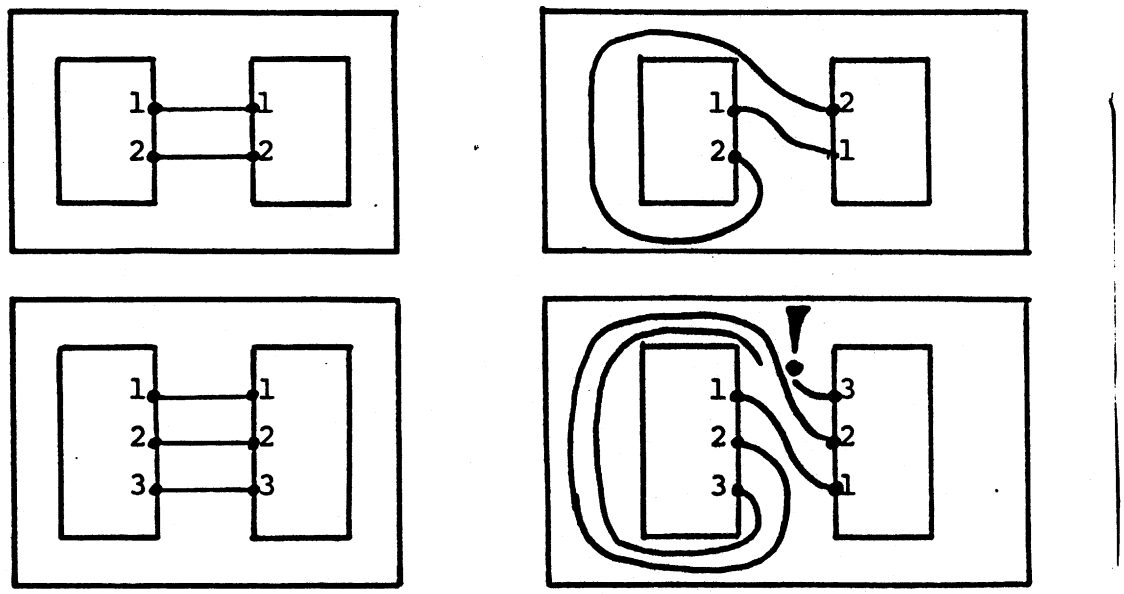


Figure IV.4: Two wires can be routed as one cable (a) or two cables (b), but three wires are forced into one cable (c and d).

The orientability test for a given cable is easy and takes constant time to perform. Essentially, all we have to do is to find which way each sequence goes relative to the inside of the module it is on, and then compare the orientations of the two sequences and see if they conform or not. The first step can be done by just looking at the first two terminals of each sequence — the rest must follow in the same direction. For the second part, we can do one of two things. We can either explicitly rotate and translate the sides of the respective sequences until they face each other (horizontally or vertically) and then check if the sequences are "parallel" across the artificial channel created that way, or we can compare the ways in which the directions of the sequences are related to the inside of the module. When traversed from terminal 1 to terminal n, the inside of the module for one sequence should be on its *right*, whereas for the other sequence the module should be on its *left*. In short, the modules should be on *opposite* sides relative to the sequences.

An explicit implementation of this test can be attained by constructing a 4×4 table indexed by the four directions of sides on which terminals can be: North, East, South, and West. Then we also define a "positive" direction for each such side; the convention is that left-to-right is positive for the horizontal sides (North and South), and top-to-bottom is positive for the vertical sides

(East and West). For example, the top sequence in Figure 1(a) is positive, whereas the sequence in the bottom left of 1(c) is negative. Now, each entry in the table tells us whether the "signs" of the two sequences for any given cable should be the same or not in order for the cable to be properly orientable. We can simply store the sign of the product at the entry, *i.e.* a '+' stands for the signs being the same, a '—' for being different. Here is the table:

	North	East	South	West
North	—	—	+	+
East	—	—	+	+
South	+	+	—	—
West	+	+	—	—

For example, the entry (East,South) in the East row and South column is '+', telling us that a left to right sequence on a south side should be matched with a top to bottom sequence on an east side, or a right to left southern sequence is matched by a bottom to top eastern sequence, as is illustrated by Figure 5. This table is computed once and for all and the lookup takes constant time (indeed, this can even be sped up due to the structure of the entries). Even if the table has not been precomputed, the computation of each entry takes constant time, applying the rule used in constructing the table.

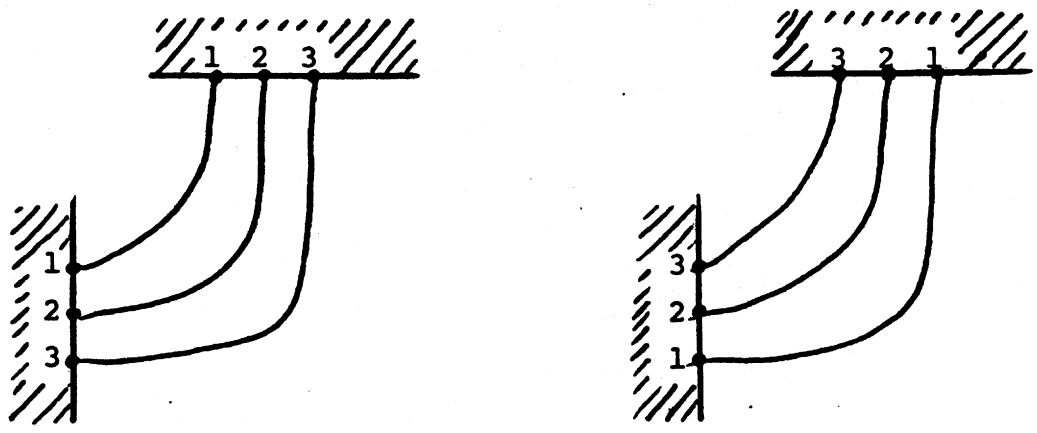


Figure IV.5: The directions of sequences on south and east sides need to have the same sign— they have to be both positive (a) or both negative (b).

If the number of wires in the cable is only two, we can first apply the above test to see if the cable is routable as one unit. If not, we know that we have to split the cable into two wires. Then, however, the reachability test has to be performed twice, since after the first wire has been

connected, we must find out whether the second one is still connectable in the new topology created by the connection made.

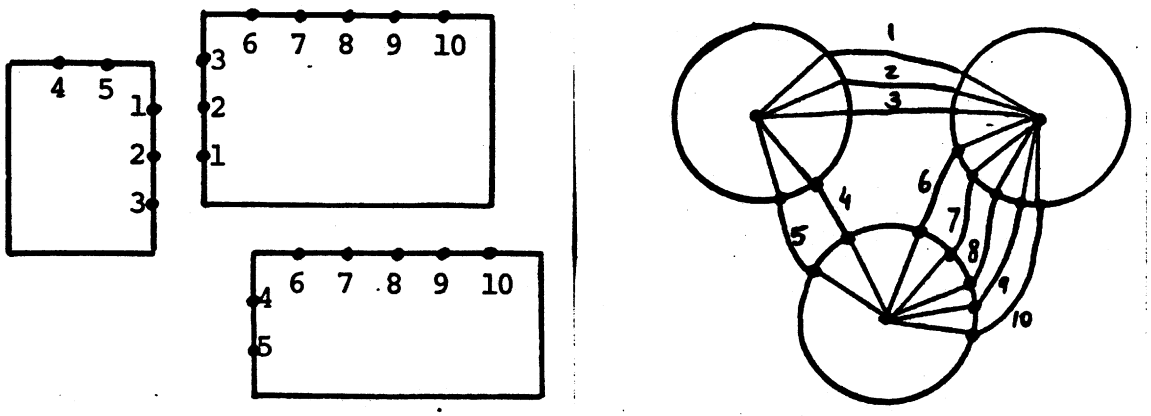
2.2. Flippable Modules

The problems concerning the routability of two wires are just a special case of Planar Routability: how do we accomodate several cables on the same chip? The layout of one cable may interfere with that of subsequent wires, and thus we need a comprehensive placement method by which individual cables can be properly oriented so they can be each routed in one layer, and at the same time not cross over (at the cable level).

The objective of making a cable properly orientable can be achieved by reflecting modules, *i.e.* flipping them over so that some "positive" sequences (in the sense of the previous subsection) become "negative" (and vice versa). Obviously, flipping a module once changes the signs of *all* sequences around it, thus some binding between sequences' orientations is imposed by each module. The second problem, of placing the modules so that cables do not get into each others' way, seems harder, since now we are trying to resolve some global conflicts. This is where the cable abstraction comes in handy. Using cables as "super-wires" we can model the overall interconnection pattern of the chip by a graph that abstracts all the necessary properties. The graph will be planar if and only if the modules can be placed and oriented in a way that makes a one-layer realization of the interconnect feasible.

Before introducing the way in which cables are used, we present the following definition of the *interconnect graph,* that is similar to the layout graph concept found in [vanLOtt73]. However, our usage and modification of this concept in this and the next subsections are more elaborate and present a different point of view on the subject.

We model each module by a "wagon-wheel" by placing a vertex in the middle of the module, making a ring out of the terminals around it, and then connect the terminal vertices to the module vertex by spokes. Each net is represented by an edge connecting its two terminals. The correspondence between a layout problem and its interconnect graph is shown in Figure 6.



(a) schematic layout (b) interconnect graph

Figure IV.6: The modules in the schematic layout of (a) are modelled by the wagon-wheel constructions in the interconnect graph (b).

Definition IV.1. For a given layout problem $\mathcal{L} = (M, N)$, consisting of a set of modules M and a set of two-point nets N with terminals on the modules' boundaries, we define the *interconnect graph* $G(\mathcal{L}) = (V, E)$ as follows: $V = M \bigcup T$ where $T = \{t \mid t \in \nu \text{ for some } \nu \in N\}$, *i.e.* there is one vertex for each module (the module-vertices) and one for each terminal (the terminal-vertices); $E = \{(m,t) \mid m \in M,\ t \in T,$ and t lies on m's boundary$\} \bigcup \{(t_1, t_2) \mid t_1$ and t_2 are adjacent on the boundary of one module$\} \bigcup \{(t_1, t_2) \mid \{t_1, t_2\} \in N\}$.

Next we use *cables* to reduce the size of the interconnect graph by modifying the given layout problem as follows: every cable that consists of 3 or more nets is replaced by exactly 3 nets — the first, last, and one of the interim original nets (lying between the first and the last net). Thus all sequences on modules sides are at most 3 terminals long. Then we use the modified layout to produce the *modified interconnect graph*. The modified graph for the layout of Figure 6(a) is shown in Figure 7. Notice that the number of vertices has been reduced from $m + 2 \cdot n$ (where $n = |N|$, $m = |M|$) to $m + 2 \cdot c$ where c is the number of cables ($c \leq n$). The number of edges in the graph becomes smaller as well.

From here on, the solution of the layout problem can be reduced to the graph planarity problem, since a layout is realizable in one layer (allowing module reflection) if and only if the corresponding (modified) interconnect graph is planar. The correctness of the reduction for the interconnect graph follows from the way modules are being modelled, which forces the order of terminals around a module to be preserved by any planar embedding of the graph. The spokes ensure that no connections are being made "inside" modules, *i.e.* neither can nets be routed

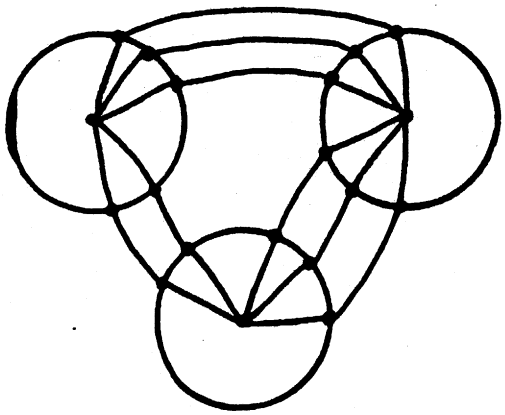


Figure IV.7: The modified interconnect graph for the schema in Figure 6.

through modules nor can one module be embedded within another. The correctness for the modified graph follows from our discussion on the orientability of cables in the Subsection 1.

Algorithms for planarity testing (and actual generation of an embedding) are known to be fast. Both algorithms presented in [Ev79, Chapter 8] (one is due to Hopcroft and Tarjan, and the other is due to Lempel, Even and Cederbaum) run in time linear in the number of the vertices of the graph. Thus the time complexity of our procedure is $O(m + c)$ once sequences have been formed.

2.3. Unflippable Modules

The planarity testing algorithms used in Subsection 2 cannot be applied verbatim to solve planar routability (PR) if the given modules are not allowed to be reflected. Now the orientation of each module relative to a global frame is fixed *a priori* (up to rotation and translation). But the wagon-wheel constructs by which modules are modelled can be embedded in the plane in two different orientations. This situation, however, can be remedied by modifying the planarity testing algorithm used. In this subsection we show how to modify Hopcroft and Tarjan's path addition algorithm [HoTa74] so as to handle such extra constraints as the orientation of wagon-wheel subgraphs.

Preprocessing. Before applying the path addition algorithm, some preprocessing is required. Obviously, a necessary condition for embedding the interconnect in the plane for the given orientation of modules is that all cables comprised of three nets or more are routable. This can be tested easily by applying the algorithm given in Subsection 2.1 for CR (we do not have to check reachability because modules can be translated freely).

Next we look at cables that are made of exactly two nets. If such a pair can be routed as one cable we do nothing at this stage. However, if the pair has to be split into two cables, the

implication of connecting both nets in the given orientation has to be taken into account. Figure 8 shows the four ways in which a pair can be connected as two separate cables. The effect on the planarity of other connections between the two modules is the same in all four cases: the (other) terminals of one module cannot be connected to the (other) terminals of the other module by any path in the plane. Nets 3 and 4 in Figure 8 demonstrate this phenomenon. Terminals whose connection is forbidden in this way are called *separated*. Our modification to the path addition algorithm will be to guarantee that no path (possibly through other modules) connects separated terminals (unless it goes through one of the edges that belong to the pair).

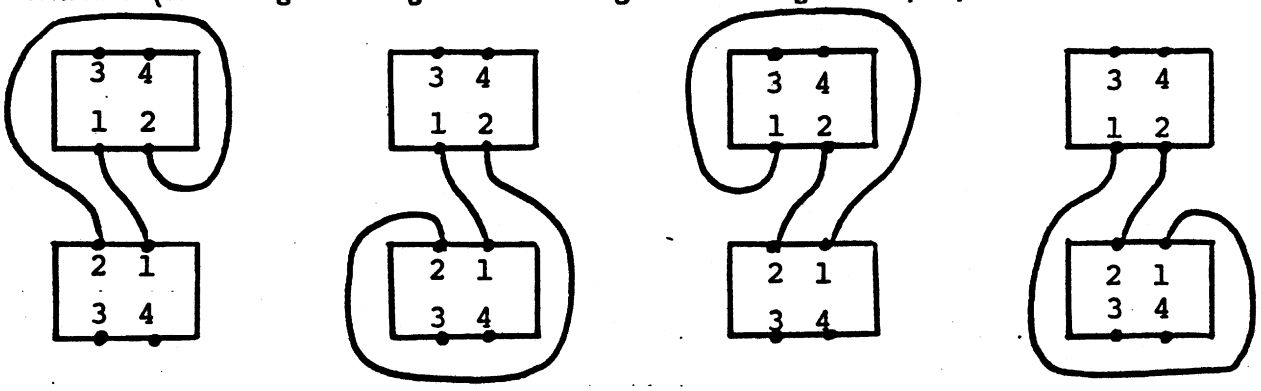


Figure IV.8: Four ways to connect nets 1 and 2, which all cut off nets 3 and 4.

For every pair that cannot be routed as a cable, we have to designate the terminals that are separated by it. We generate a pair of complementary labels, l and $\bar{l}$, for every problematic pair of nets.

Modifications to the path addition algorithm. Now we can start running the path addition algorithm. Every time an edge $e = (u, v)$ is traveresed during path generation the labels from the tail, u, are added to the labels of the head, v. There is one important exception: an edge that belongs to the pair of nets that generated the labels pair $(l, \bar{l})$ *does not* propagate either of these labels.

The planarity *testing* is modified as follows: every time a net edge (as opposed to a wagon-wheel edge) is traveresed, we check if a conflict has occurred: if we try to add $\bar{l}$ to a set containing the label l the planarity test fails. (This failure condition comes in addition to the failures which may arise from the original parts of the algorithm.)

Correctness. During the path addition algorithm every edge is traversed once. By the definition of the label propagation rule, the only way separated terminals are connected via a path* is if they use a net edge that caused their separation, but not through any other net edge.

Complexity. The modifications required in the path addition algorithm cause only constant overhead per step. We have to assume, however, that set operations such as union and membership can be done in constant time; this can be achieved if a membership vector is used to represent the sets. The size of the sets that have to be kept is the number of problematic pairs, which is persumably small. The presprocessing that is required is linear as well.

All in all we have shown

Theorem IV.1. Planar Routability (PR) can be solved in linear time.

3. Detailed Routing

3.1. Routing in a Simple Rectilinear Polygon

Given an instance of Detailed Routing in a Simple Rectilinear Polygon, we start off by checking whether the connections specified by the nets are at all realizable in one layer from a topological point of view, *i.e.* whether the interconnect pattern is planar when constrained to lie within the polygon (ignoring minimum spacing rules). We could have used a test similar to the general test suggested in Subsection 1.2, but here we can use a simpler method. What we have to check is whether the terminals as they appear along the boundary of the polygon match without intersecting. This is analogous to checking whether a set of parentheses is properly balanced, as formulated in the following algorithm:

1. Initialize the stack S to be empty.
2. Cut the boundary of the polygon at any (single) point and straighten it out.
3. Scan the list of terminals from left to right. For each terminal compare the net number (or any other form of a net id) to the number at the top of S. If they are equal, pop S, else push the number on S.
4. If at the end of the scan S is empty, the nets are properly nested. Otherwise, they are not realizable in one layer.

* Notice that a path in the graph may be comprised of parts of several paths that have been added during the algorithm.

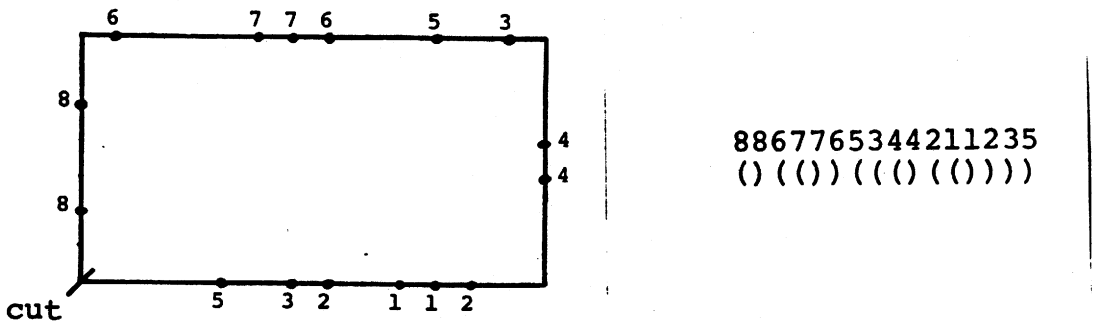


Figure IV.9: The correspondence between planar routing in a polygon (a) and parentheses expressions (b).

Correctness. We interpret the first occurrence of a terminal in a net as an *opening* (left) parenthesis, and the second occurrence* as a *closing* (right) parenthesis. Regardless of where the boundary has been cut, the interconnection pattern is internally planar if and only if the parentheses in the expression (obtained by the above interpretation) are properly balanced. Nets must be nested within each other or be mutually exclusive in their span in order to allow a planar (non-crossing) realization, which is the same as requiring that a set of parentheses be balanced. This correspondence is demonstrated by Figure 9, parts (a) and (b).

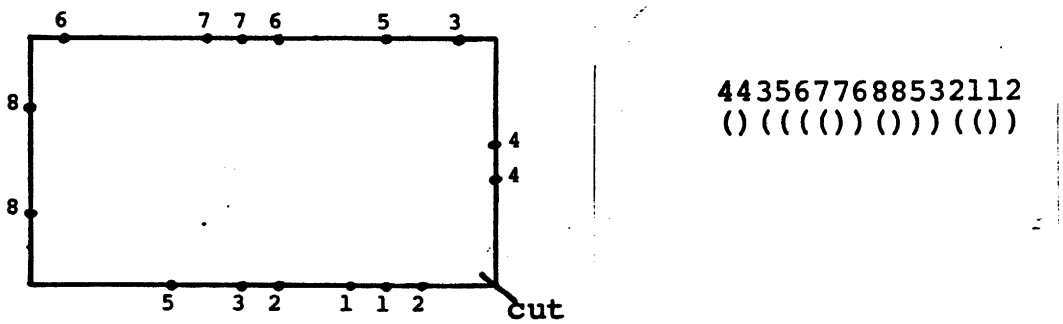


Figure IV.10: Changing the cut position does not affect the validity of the parentheses balancing test.

The position of the cutting point is immaterial since the area enclosed by the polygon can be viewed as a half-plane for purposes of planarity checking; thus it does not matter how this half-plane is spanned relative to the original setup. This is demonstrated by Figure 10, which shows a different cut than the one used in Figure 9 without changing the outcome of the test.

* Remember that we are dealing with two point nets only. Multipoint nets can be treated by duplicating the internal terminals, thus inserting a ")(" in the parentheses expression.

From here on we shall discuss detailed one-layer routing in a rectangle. The techniques developed to solve both the routing and the routability problems for a rectangle are generalized later to the case of a simple rectilinear polygon.

We now turn to the question whether the area of the rectangle is enough to realize the routing of the nets relative to a given wiring model. As has been pointed out in previous chapters, there is a quantitative difference (in terms of time complexity) between the *routability* question and an actual routing algorithm, *i.e.* it is sometimes faster to check whether a set of nets is routable within a given area than to produce the actual paths taken by the wires. For the problem of river routing across a channel, for example, we saw that testing routability takes time $O(n)$ for n nets, whereas producing the layout may take time $\Omega(n^2)$. In what follows we show that the situation for river routing in a polygon is similar. It is easier to test for routability than to produce a layout.

For purposes of clarity, we shall start off by providing an algorithm that river routes nets in a rectangle whenever possible.

Use the template given in the algorithm for planarity checking (without straightening the boundary): whenever a net is being popped off the stack, simply *route* it! The routing is done in a "greedy" fashion relative to the contour — the router stays as close as possible to the boundary by initially routing nets along the original boundary (for "innermost" nets), and then staying as close as possible to the contour formed by previous nets towards the boundary. This routing strategy is demonstrated by Figure 11.

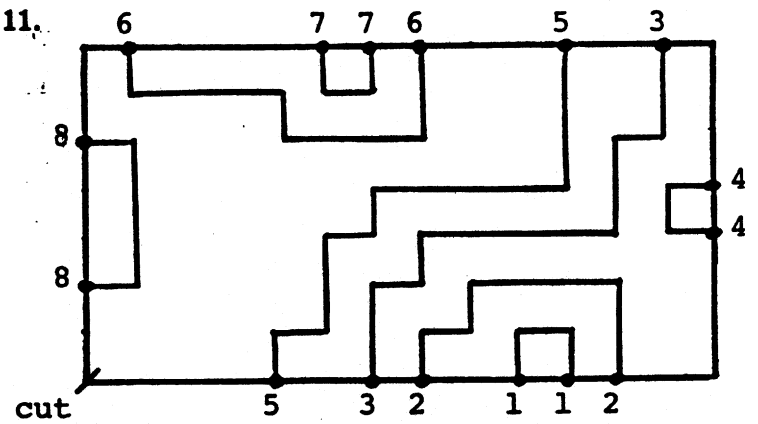


Figure IV.11: Routing the example of Figure 9(a). The cut is in the SW corner.

This procedure takes time that is proportional to the number of straight (or elementary) wire segments that are produced; this number can be as large as $\Omega(n^2)$ in the rectilinear model, as shown, for example, in Figure 12 (where n is the number of nets to be interconnected). The worst

case time complexity of the algorithm is $O(n^2)$, matching the existential lower bound, thus it is optimal from this point of view. However, it may produce results that are suboptimal in terms of wire length and number of jogs.

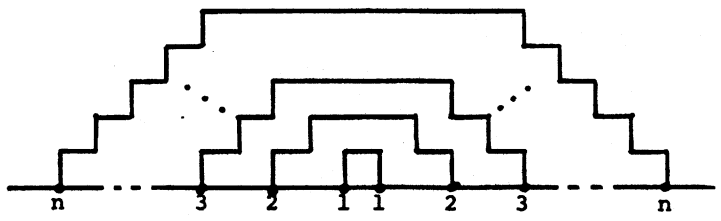


Figure IV.12: The greedy routing rule may produce $O(n^2)$ segments even when all terminals are on one side.

In fact, this algorithm is just one member in a family of algorithms obeying the following *greedy routing* rule:

> "Each net is routed when all* nets between its terminals along the boundary (according to the linear order) have already been routed."

The argument for the sufficiency of such a rule is quite obvious: all nets are routed in a way that minimizes their mutual interference. Each net is routed as tightly as possible relative to the area requirements of the nets that have been routed so far, and as far away as possible from any subsequent net. The order of routing is dictated by the planarity — up to mutual exclusive nestings — for any given cut. The position of the cut is immaterial: although it does dictate the distinction between "opening" (left) and "closing" (right) parentheses as far as terminal pairs are concerned, and so also makes the association between a net and the side(s) of the boundary it is supposed to be routed closest to, there is no difference between one cut and another as long as the association of nets with boundaries is consistent between nets for each cut. Due to the fact that wires are routed according to the greedy rule, no area is taken up unless it is required by the minimum spacing conditions, so the side the net is associated with does not affect the routability of the whole area. Figure 13 shows how the example of Figure 11 will look if the cut point is moved as shown.

* This condition may be vacuously true, *i.e.* it holds when there are no other nets between the terminals of the current net.

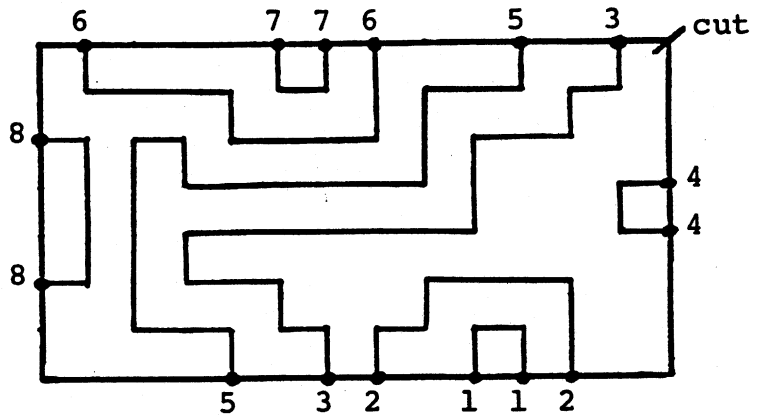


Figure IV.13: Routing the example of Figure 9(a) using a northeast cut.

This brings us to the fine point concerning failure to route an area. Since the greedy routing rule is sufficient, at least one net will be unroutable if the area is not large enough to accomodate the pattern. The identity of this net *does* depend on the location of the cut, as can be seen in Figure 14.

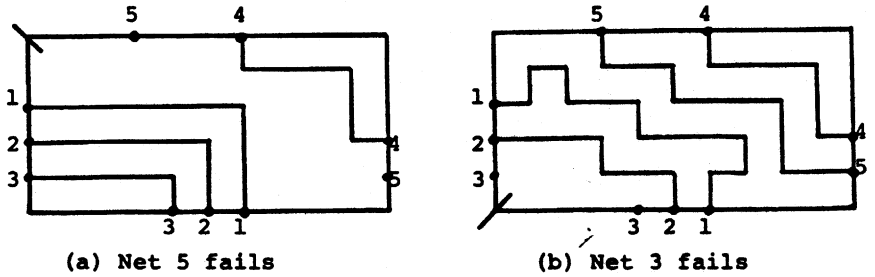


Figure IV.14: Different nets manifest unroutability depending on the location of the cut.

Now we turn to the question of routability. Some terminolcgy is due at this point. We classify nets according to the relation between the sides their terminals lie on. If both terminals are on the same side of the rectangle, the net constitutes a *single-sided connection.* If the terminals are on adjacent sides, we say the net is a *corner connection.* If they are on opposite sides, it is a *cross connection.* Also, for purposes of presentation, we label the four sides of the rectangle by definite names — north, east, south, and west. The corresponding connections inherit the appropriate names, e.g. "north-east connections" (the orientation is arbitrary, of course, but once it has been set it remains fixed). Notice that although single-sided connections and corner connections can occur at all possible locations in one instance of the problem, cross connections can either go north-south or west-east, but not both (as anyone who plays Hex or Bridge-It can tell you). This is trivially due to the planarity requirement on the interconnect pattern (see Figure 15).

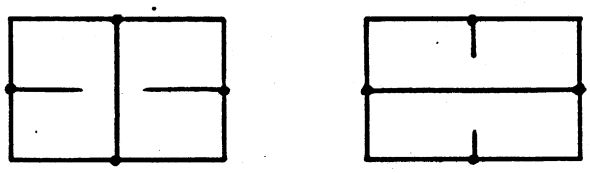


Figure IV.15: Cross connections can be of one kind only.

From here on we restrict ourselves to the square grid wiring model. In addition, as you may have already noticed by looking at the figures, we do not allow wiring on modules' sides (as we did for river routing across a channel); thus, the mathematics that we borrow from Chapter III has to be slightly modified by adding or subtracting 1 when moving away from the boundary. For example, if the separation between the sides of a channel is measured by t with such boundary conditions, *i.e.* the number of tracks available for routing is $t-1$, the routability conditions become $a_{i+t-1} - b_i \geq t-1$ and $b_{i+t-1} - a_i \geq t-1$.

The technique we use here to obtain the routing requirements without generating all the routing details is a generalization of the method used for river routing across a channel. Recall the geometric interpretation of the river routing constraints as 45° lines drawn from terminal positions towards the other side of the channel. These lines told us how other terminals had to stay clear from the source of the "ray" to accomodate the wires around its net. Since the interconnection pattern was known (all nets went in order across the channel) and monotonic wiring was sufficient, we could make all lines the same length (as a function of the separation) and compare each endpoint statically to the rest of the channel. Here the structure is more complicated, and we have to dynamically change the length of the "ray" we draw in order to establish the routing requirements around the current net relative to the other nets.

Let us start with the simplest case — generating the boundary around single-sided connections. Here there is no problem of routability because the connections can always be realized (assuming the rectangle above the side has unbounded height), and all what we seek is the smallest area required for routing. The output of our procedure will be a description of the routing contour that is necessitated by the given configuration, *i.e.* the path closest to the boundary that can be taken by the outermost nets. As any path, the contour can be described either by its segments or its corners. We shall produce a representation by corners. Due to the way in which the spacing rules affect the wiring, the skeleton of the contour is generated by its convex corners, *i.e.* corners that protrude from the perimeter of the rectangle inwards. Convex corners that arise

from the spacing calculations are called *markers*. We shall see now how to set the markers and generate the path using them.

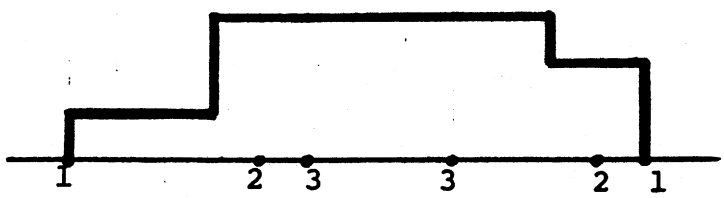


Figure IV.16: The contour of a set of nested single sided connections.

For sake of presentation we give the details of how to construct the contour along a south side that is being scanned from left to right, as shown in Figure 16. We assume that each terminal has two pieces of information associated with it (formulated in terms of parentheses expressions): whether it is an opening parenthesis or a closing one, and its depth (or nesting level) in the expression*. Markers corresponding to a south side are either northwestern or northeastern (convex) corners. The west end of the south side is the origin $(0,0)$. Markers are set according to the following rules:

(1) For an opening terminal at $(x,0)$ and depth i, establish a northwest marker at $(x-i+1,i)$.

(2) For a closing terminal at $(x,0)$ and depth i, establish a northeast marker at $(x+i-1,i)$.

The geometric interpretation of these rule is simple. Imagine 45 degree "rays" that emanate from one row (of the grid) above the side at terminal positions, *i.e.* the ray for a terminal at $(x,0)$ starts at $(x,1)$. The length of the ray is one less than the depth of the net, and rays corresponding to opening terminals go to the northwest whereas those corresponding to a closing terminal go to the northeast. Figure 17 exemplifies the procedure. Each marker in the figure is tagged by the number of the net that generated it. Northwest (NW) markers were generated by opening terminals, and northeastern (NE) ones by closing terminals.

* This information can be easily generated on the fly.

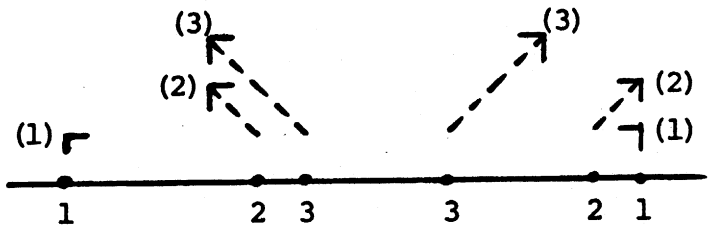


Figure IV.17: Determining the contour for single-sided connections using markers. The net numbers correspond to their nesting level.

Intuitively, NE markers "guard" the area that is immediately below and to the left of their position, and similarly NW markers "guard" the area below and to their right. All we must find out is how far these areas go laterally, *i.e.* how markers are to be connected in order to form the contour. Notice that markers generated by terminals of the same net are both at the same height (have the same y-coordinate). We associate with each net the rectangle formed by the line connecting its markers, their projections on the side and the portion of the side bounded by the projections. The routing contour is defined by the perimeter (excluding the side itself) of the union of these rectangles. Performing the union is too costly, however, and fortunately can be avoided. Essentially, what we do is trace the markers as they are being produced and connect them according to certain rules.

The contour tracing rules have to accomodate two subtle phenomena. Net 1 in Figure 18(a) reveals a point concerning the detection of "gaps" between opposite markers: "bumps" are caused by deep nestings that are far from each other, but some nets span the side between the bumps. We have to make sure that at any point (x-coordinate) the contour cannot be closer to the boundary than the depth of the parentheses expression (by a simple density argument). Secondly, as can be seen in Figure 17, some markers are right above other markers of the same kind that were generated before them (such as 3 above 2), and more importantly — some northwestern markers end up to the west of northeastern markers that belong to terminals on their east (such as 2 and 5 in Figure 18(b)) and vice versa. Such "dominated" markers must be ignored so that the correct contour will be produced.

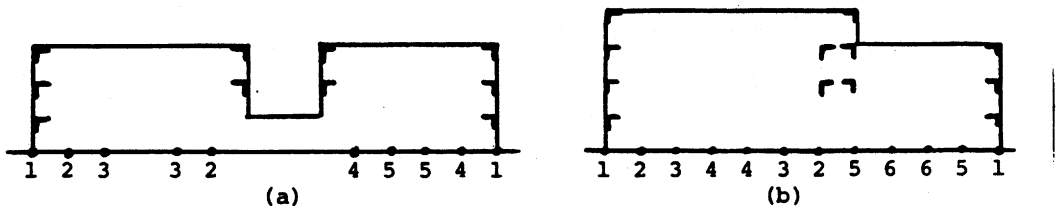


Figure IV.18: Problems with marker tracing: gaps (a) and domination (b).

The tracing algorithm which follows scans terminals from left to right. Markers are generated according to the rules (1) and (2) by the iterator next_marker that returns the triplet (x, y, t), where x and y are the coordinates, t is the type (NW or NE). The algorithm outputs the corners of the contour, but notice that some outputs may be erased.

```
x_max^NE ← 0;   x_max^NW ← 0;
for (x, y, t) ←next_marker do
    if t=NE then do
        if x = x_max^NE then erase last corner
                        else output(x, y) fi;
        output(x, y − 1);
        x_max^NE ← x
        od
    else comment t=NW; do
        if x ≤ x_max^NE then let (x*, y*) be a corner that was
                             generated by a NE marker such that
                             x* = min{x' | x' ≥ x}
                             and y* = max{y' | (x*, y') ∈ contour}
                        if y* ≤ y then do
                            erase all points from (x*, y*) to the end;
                            output(x, y*);
                            output(x, y);
                            x_max^NE ← x
                            od
                        else ignore (x, y) fi
                    else comment x > x_max^NE; do
                        if x = x_max^NW then erase last corner
                                        else output(x, y − 1) fi;
                        output(x, y);
                        x_max^NW ← x
                        od
        fi
    fi
end_for
```

The correctness of the rectangle union procedure can be deduced from Tompa's analysis in [To80]. In order to show that the above algorithm realizes it, we must study the marker generation more carefully. First, we observe that markers of the same kind are being produced in *ascending* order relative to their x-coordinate:

Lemma IV.1. If $x_1 < x_2$ are the x-coordinates of two opening (closing) terminals t_1 and t_2, then $x_{m_1} \leq x_{m_2}$ where x_{m_i} is the x-coordinate of the NW (NE) marker generated by t_i.

Proof. We shall prove the lemma for opening terminals since the case for closing ones is analogous. Number the opening terminals from left to right as they appear, so that x_1 is numbered i and x_2 is numbered j, where $i < j$. Denote $k = j - i$. Due to the unit spacing design rules, $x_2 \geq x_1 + k$. Also, if t_1's nesting depth is d_1 and t_2's is d_2, then $d_2 \leq d_1 + k$. Thus,

$$\begin{aligned} x_{m_1} &= x_1 - d_1 + 1 \\ &\leq (x_2 - k) - (d_2 - k) + 1 \\ &= x_2 - d_2 + 1 \\ &= x_{m_2}. \end{aligned}$$

∎

The second observation is that the y-coordinates of two markers generated by consecutive terminals differ by exactly 1 if the terminals are of the same kind (both opening or both closing), and are the same if the terminals are of different kinds.

Lemma IV.2. If $x_1 < x_2$ are the x-coordinates of two consecutive terminals t_1 and t_2, and y_{m_i} is the y-coordinate of the marker generated by t_i, then

(i) $y_{m_2} = y_{m_1} + 1$ if both terminals are opening;

(ii) $y_{m_2} = y_{m_1} - 1$ if both terminals are closing; and

(iii) $y_{m_2} = y_{m_1}$ if the terminals are of different kinds.

Proof. Trivial, by the definition of markers and properties of balanced parentheses expressions. ∎

From these two lemmas we can deduce that the only kind of "crossover" between a sequence of NE markers and one of NW markers is as shown in Figure 19 — each sequence is monotonic. Any other pattern will violate one of the two lemmas. Thus the contour is generated correctly, without creating degenerate (straight) corners.

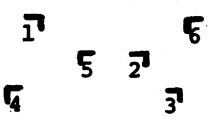


Figure IV.19: The only way markers can cross is as two monotonic sequences.

The time complexity of the whole procedure is $O(n)$, where n is the number of nets. The only part that seems difficult is finding (x^*, y^*). We exploit the fact that markers of the same

kind are generated with consecutive y-coordinates, and that the only type of crossing is between two monotonic sequences as shown in Figure 19. The idea is to store NE markers in an array indexed by their y-coordinate (array cells may be reused later), and whenever a crossing NW sequence is generated, the crossing is found by a single test. To make sure the complexity is linear, each array cell must contain a pointer to the maximum y that has the same x position, which may be different from the y of the stored marker.

So far we have proven:

Theorem IV.2. The routing contour required by n two-point nets whose terminals are on a single straight line can be computed in time $O(n)$ for the one-layer rectilinear wiring model.

The next step is to deal with corners. Notice that a corner connection may have single-sided connections nested within it, as shown in Figure 20, but not vice versa. Thus single-sided connections need not be dealt with any further.

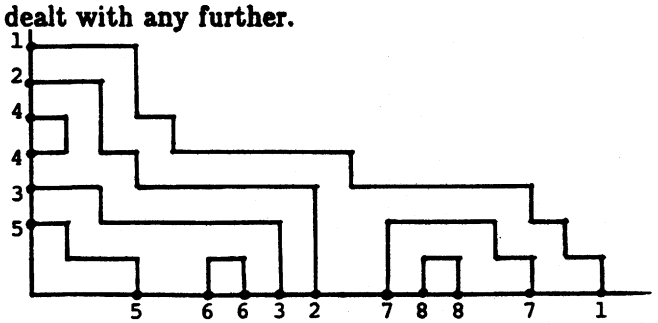


Figure IV.20: Corner connections (1, 2, and 3) with-single sided (and other corner) connections netsed within them.

For concreteness, we describe the procedure for a southwest (SW) corner. Unlike single sides, corners may not always be routable. First, single-sided connections coming off orthogonal sides could interfere with each other, as in Figure 21. Second, connections to terminals may be blocked by narrow passages between the single-sided contours.

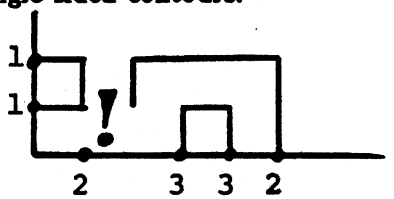


Figure IV.21: Interference between single-sided connections coming off adjacent sides.

To solve these interference problems we maintain a separate contour for each side, and then check whether the contours intersect. Thus all single-sided connections are processed first, setting up contours for subsequent analysis. We also process those corner connections that do not have single-sided connections embedded in them, which is an easy extension to the single-sided algorithm, and in fact, none of the complications arise. The contour of these innermost corner connections can be associated with either side of the corner.

The essence of the procedure for testing the routability of corners is finding out whether the wires that must be connected to the sides have enough room to pass through the "straits" formed by the single-sided countours. The test we are shall devise applies not only to making corner connections *per se,* but also to the ability of terminals that are on the sides of a corner to "get out of the corner" and be connected to their mates regardless whether the mates reside on the other side of the corner or on another side of the polygon altogether. However, for sake of clarity, we start off by restricting ourselves to connections that are all on the corners in order to understand the general phenomena that will be utilized when constructing the overall test for a rectangle.

We define the routability problem for an open corner as follows. Given a quadrant with terminals of two-point nets on its sides, can the nets be connected using one layer subject to the minimum spacing requirements?

To solve the routability problem for an open corner, we devise a uniform way to *(i)* ensure that all "straits" are wide enough to accomodate the nets that have to go through them, and *(ii)* check whether the contours themselves collide. The technique is to draw forty-five degree "rays" from all convex (protruding) corners of single-sided contours and find their intersection with the rest of the contour. Recall that the contour consists of original sides as well as pieces generated by single-sided connections.

Each ray is trimmed at its first intersection with the rest of the boundary. If the ray hits the boundary from the "inside" then we know that two contours collide and the test fails. If, on the other hand, the ray has positive length, we compute how many wires can cross it. The *crossing test* for a ray succeeds if this number is not smaller than the number of nets that *must* cross it.

Lemma IV.3. An open corner is routable if and only if the crossing test succeeds for *all* forty-five degree rays emanating from convex corners of single-sided contours.

Proof. The necessity of the tests is obvious: if any of the rays is over-congested no wiring can take place across it. The sufficiency is a consequence of the sufficiency of the greedy routing rule. To realize that, we must examine the structure of the interconnect pattern left after single-sided connections have been processed.

Each wire must go around at least one contour that was generated by single-sided connections. We can group together terminals of corner connections according to the *gaps* between* contours of single-sided connections in which they reside. Notice that all terminals in the same gap are connected to the other side; in particular, if there are no innermost corner connections, the terminals of corner connections that are closest to the corner itself lie all on one side.

* We consider the south-side segemnt to the east of the eastmost terminal and the west-side segment to the north of the northmost terminal as gaps as well.

The areas between neighbouring single-sided contours (and outside the outermost contours) can be perceived as channels *through* which the nets connecting to the terminals in the gap have to go. Since both parts of the neighbouring contours are monotone with respect to this channel, the analysis given in Section III.1 regarding the sufficiency of the greedy routing rule applies here as well. In essence, each convex corner of a single-sided contour is present due to a terminal on the side. The ray drawn from this corner is an extension of the ray used in the proof in Section III.1. The only difference is that for corner connections there is no simple characterization of the form the wire is going to take in terms of the overall terminal specifications. We need to process single-sided connections first to set the framework so the analysis can be used. ∎

Remark. In fact, only rays that are being drawn towards *the other side* of the corner matter. That is, when dealing with a SW corners, we have to consider only rays that emanate from NW markers on the contour of south single-sided connections, and SE rays that emanate from the west connections. The contour of innermost corner connections can be ignored altogether (as far as the generation of rays is concerned).

Complexity. The number of wires that can cross a forty-five degree line is one less than (the floor of) the length of its rectilinear projection. To find the number of wires that must cross the trimmed ray, we count the number of corner connections that have one terminal on each side of the line. This can be done efficiently by maintaining a direct access data structure (such as an array) to record the contour as it is being generated and associating with each entry the identity of the relevant corner connections as they are being scanned.

Now we are ready to introduce the routability test for a rectangle. We start by constructing all the single-sided contours (including contours of corners that do not have single-sided connections embedded within them). The next step is a generalization of the corner routability test. Three rays (rather than one) are being drawn from each corner, and also rays are being generated by terminals.

The three rays that are generated at each convex corner of a contour go outwards relative to the inside of the contour in the following directions: vertically, horizontally, and in a 45° angle. The first and second rays are extensions to the contour itself, and the third bisects the right angle created between them, as shown in Figure 22. From each terminal we draw now two forty-five degree rays — one to each direction.

The test for routability is the same as in the corner case: all trimmed rays must be longer* than the number of nets that have to cross it. A ray that starts from within the contour of another side has negative length, thus the test as stated covers the contour intersection problem as well.

* We do not subtract 1 from the length in this statement, so that is why we just say "longer".

The necessity of the three rays is due to the interaction between single-sided contours that are generated on opposite sides. Forty-five degree rays are not guaranteed anymore to be trimmed by a perpendicular side, thus we need an explicit density test between opposite contour. This is exemplified in Figure 23, where the forty-five degree rays miss the contour on the opposite side, but the passage between the contours is too narrow to permit four nets to be routed across it.

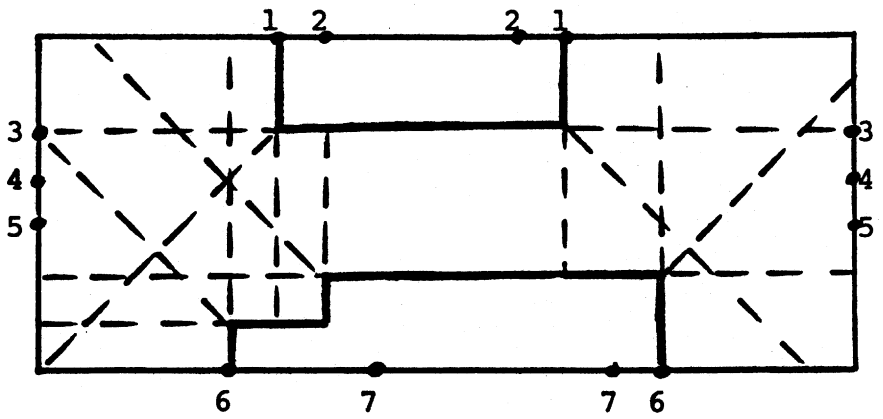


Figure IV.23: The necessity of rectilinear rays is demonstrated by the narrow passage between the single-sided contours.

The sufficiency of the test is shown by the same argument as for open corners. Each triple of rays ensures that the contour-corner can be cleared by all wires that have to pass along it if a greedy routing rule is used.

The argument concerning rays that emanate from terminals that belong to cross and corner connections also is similar to the proof used for river-routing across a channel (Section III.1). Now, however, we cannot assume any routing fashion insofar as monotonicity is concerned. Also, we have to account for connections that are very close to corners on both of its sides (this is possible now). Therefore rays must be drawn in both directions.

Notice that if the rectangle happens to have only cross connections, we end up applying the river routing test twice — once for a left to right enumeration of terminals, and once for the other way around. Obviously, if one set of tests succeeds so will the other, but the overkill is required to cover the general case.

The linearity of the general test is shown by the same argument as for an open corner. So far we showed

Lemma IV.4. Planar routability of n nets in a rectangle can be checked in time $O(n)$.

When routing in a rectilinear *polygon*, rather than a rectangle, the corners of the polygon have to be treated as contour-corners, generating three rays each. If the polygon is known to be simple, all routability tests can be done in linear time. So all in all

Theorem IV.3. Detailed planar routability in a simple rectilinera polygon (DPRSP) can be decided in linear time.

3.2. Routing around Modules

The problem of detailed routing around modules when no homotopies of wires are given (DR) is NP-complete. In this subsection we show how to modify a recent result of Kramer and van Leeuwen [KrvanL82] to obtain the claimed statement. The intractability of the problem is due to the possibilities of routing wires around modules in more than one way. Unlike the situation in Section 2, where planarity was the only concern, here the spacing requirements play a key role. Routing a wire using a certain path may render other wires unroutable due to the consumption of routing area by the first wire. Thus some global decisions need to be made in a way that is so typical of most known NP-complete problems.

The two problems discussed in [KrvanL82] concern one-layer routing of two-point nets ona square grid. In the first problem (*"Routing"*) routing is done in the absence of modules altogether. In the second problem (*"Obstacle Routing"*) modules are allowed as part of the problem specification, but they are viewed only as *forbidden areas* rather than pieces of logic with connections on their boundaries. Thus all terminals are grid-points that can be routed from in all four directions (unless one of the immediate neighbours is occupied by a terminal of a different net). This deviation from our notion of terminals as residing on modules' boundaries can be remedied easily by providing a simple construct for local replacement [GaJo79, Section 3.2.2] to be used in our transformation.

The transformation can be made from either of the problems discussed in [KrvanL82]. We prefer here to use the problem in which modules (obstacles) are present. Notice that in all the constructions of the proof in [KrvanL82], a path leaves a terminal in one of two opposite directions. Thus all arguments can be retained if we construct a layout in which the paths are constrained a priori to leave a point in one of two opposite directions (or even in one of three directions). This can be done by using the replacement of Figure 24: terminal P in the original construction is replaced by terminal P' on the side of a module that is parallel to the direction in which the paths leaving P go in the original construction. The point P has to coincide with P'', the immediate neighbour of P', rather than with P' itself.

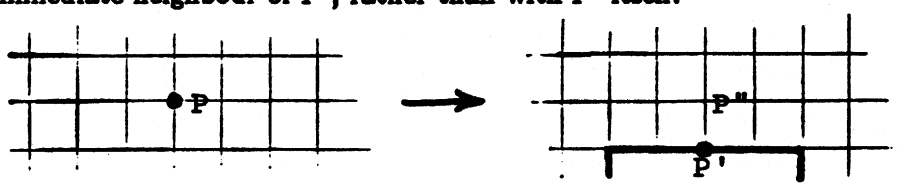


Figure IV.24: A pin in the Kramer and van Leeuwen construction can be replaced by a terminal on a module's side.

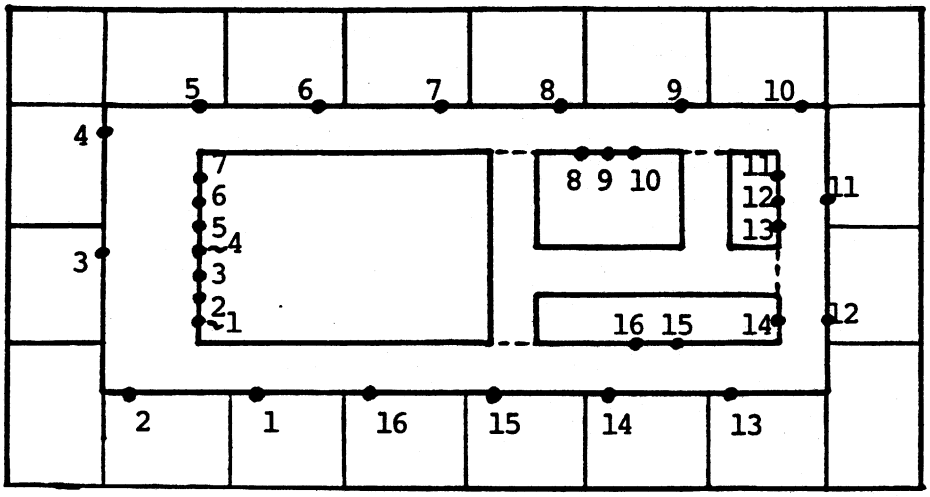


Figure IV.25: River routing to pads.

Since most terminals in the construction of [KrvanL82] are located next to modules (obstacles) anyway, and their routing direction is parallel to the side of the module, this replacement can be performed by just moving P to the boundary. In the few cases where this is impossible, we splice a new module into the layout, thereby expanding the grid by the necessary dimensions without affecting the area available for routing. Thus the feasibility of a solution remains invariant under this transformation, proving

Theorem IV.4. Detailed planar routing is NP-complete.

The major question that remains open concerning detailed routing is that of finding the detailed routing when the homotopy for all wires is given a priori (DRH). Then the decision of how to route around the modules topologically is solved, but we still have to find exact paths for the wires. When no modules (holes) were present, nets could be routed using the *greedy routing* rule defined in the previous section. Now, however, there is no unique contour with which nets can be associated to start with: it is unclear "how long" a wire has to saty close to the side of one terminal and when to start moving towards the side of the other terminal in the net (not to mention sides that the net passes by on the way).

This problem manifests itself even in the simple situation of a *donut router,* which is typical for routing to pads, as can be seen in Figure 25. Johannsen [Jo81, Appendix 3] provides a heuristic for solving the problem that is based on the greedy routing rule for a channel, but his strategy does not produce optimal results.

4. Conclusions

In this chapter we have introduced the *cable* abstraction for planar routing. We saw that cabels are useful for two reasons. First, they reduce the complexity of the combinatorial problems that arise when modelling the orientability and placement problems. Second, keeping wires that

have to carry parallel signals together this is desirable for delay computation purposes (although it may be a bad idea when the routing is too tight or the number of wires is large due to cross talk).

Later, when dealing with detailed routing in a simple polygon, we realized that single-sided connections are the source of many complications in the routability test. Although the problem of generating the routing contour around such connections (in the absence of othe requirements) is easy, the conditions they set up for the rest of the test are irksome. It may be useful if a global routing heuristic doesn't generate single-sided connections at all, thus simplifying the routability test for each channel considerably.

Chapter V

Routing Two-point Nets Across a Channel

At this point in the thesis we forsake the planar paradigm that has been explored so far, and start to investigate routing patterns that require more than one layer. From here on, wiring models will be used depending on the interconnect topology and layer assignment method used.

In Chapter III we started off our study of one-layer realizations by looking at river-routing across a channel. In this chapter we route across a channel, but discard the planarity assumption. The resulting interconnect pattern is *routing two-point nets across a rectangular channel.*

Why do we restrict ourselves to dealing with channels in the first place? Routing (in its full generality) is believed to be intractable. Hashimoto and Stevens [HaSt71] proposed the channel paradigm to battle the overwhelming complexity of the problem. Since then, many authors have described heuristics to handle signal routing in the channel framework ([Ak72], [Hi74], [Deu76], [Pr79], [Bar81], [SouRo81], [MaKu82], [YoKu82], [Riv82], and [RiFi82], to mention only a few). Early work was motivated by applications to printed circuit board (PCB) layout, but recent advances in VLSI technology have prompted a renaissance of the field.

Recently the theory of computing community began to pay attention to the channel routing problem. A better understanding of the fundamental phenomena that occurr when it comes to detailed, actual design has been gained in the past two years by looking at restricted, albeit not impractical, instances of channel routing ([BrRi81], [DKSSU81], [LaP80a], [LeiPi81], [LloRa81], [RBM81], and others). Most notably, routing of *two-point nets* across rectangular channels has been studied intensively. In such channels, terminals occur only on two opposite sides and each signal is comprised of only two terminals — one on each of the opposite sides as in Figure 1. Understanding the routing problem in even such a limited context has already proven instrumental

when analyzing more general situations, and will no doubt continue to play a key role in the future.

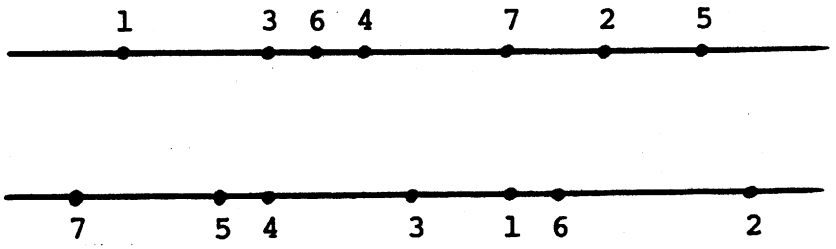


Figure V.1: Two-point nets across a channel.

In this chapter we present a digest of results concerning this routing situation — some previously known and some new. (All results stated as theorems are new.) In Section 1 we refresh some of the terminology from Chapter II and introduce some terms specific to this chapter. In Section 2 we first survey known results on minimizing channel width, and then take up the width issue in the framework of the newly proposed via-free model. Section 3 introduces monotonic versus nonmonotonic routing strategies, Section 4 discusses jogging, and Section 5 deals with the resolution of conflict cycles using monotonic routing. We conclude this chapter by listing some relevant open problems.

1. Terminology

When routing two-point nets across a rectangular channel, we are given two parallel line segments* whose relative lateral positioning is fixed. These line segments are called the *sides* of the channel. On each of the two lines there are n terminals labeled by the integers $1, \ldots, n$. The number a_i denotes the x-coordinate of the terminal on the top side whose label is i, and similarly the x-coordinate of the ith bottom terminal is b_i. The ith signal-net is represented by the pair $N_i = (a_i, b_i)$. The *width*, sometimes called the *separation*, of the channel is the distance between the sides. The channel itself is perceived as the (infinite) stripe bounded by the sides.

When routing in a channel, wires are not allowed to lie on its sides, contrary to the convention of Chapter III. Therefore the segment attached to each terminal is always perpendicular to the side, which is vertical according to our convention. Each horizontal grid line lying within the channel is called a *track*. Segments of vertical lines bounded by the sides are called *columns*. The width of the channel is one more than the number of tracks, and its *length* is the difference between the x-coordinates of the leftmost and rightmost columns used for routing. A horizontal path segment is called a *jog track* and a vertical one is a *jog column*.

* These lines are typically sides of modules enclosing the channel between them.

2. Width

The primary concern in routing across a channel has traditionaly been to reduce the number of tracks to a minimum. Most of the time no lateral movement of the sides is allowed, so the probelm is one-dimensional in a sense and is perceived as a routing problem rather than a placement problem. When the sides are allowed to move laterally as well—they have to remain parallel—the *offset* problem that is generated is indeed a placement problem.

Other optimization criteria, such as minimizing the total wire length, or minimizing the longest wire, have not received much attention in the literature. The interrelation between these criteria, especially in the context of the offset problem, is also quite interesting, but has been neglected all the same.

In this section we shall confine ourselves to the width problem in the static, one-dimensional setting. (The generalizations to the offset problem and other optimization criteria will be discussed in a forthcoming paper.) First we describe the state of the art in the study of the width problem in exisitng wiring models, and then present some new results on the layer minimization and widths problems in the via-free model. Throughout this chapter we shall keep the width problem in mind, relating our results to this important criterion.

2.1. Minimising Width in Conventional Models (summary of results)

River routing across a channel was discussed in Chapter III. The terminals on the top side, $a_1 < \ldots < a_n$, are in the same *order* as the terminals on the bottom side, $b_1 < \ldots < b_n$ (see Figure 8). Thus no crossover between wires is mandated and a via free realization (in one or more layers) is feasible. Efficient solutions for minimising the width of the channel in this situation exist and have been studied extensively in [DKSSU81], [LeiPi81], and [Bar81]. If routing is rectilinear the solution can be found in time $O(n)$, which is optimal. The channel is scanned from left to right and the routability conditions of the form $a_{i+t} - b_i \geq t$ and $b_{i+t} - a_i \geq t$ are being tested on the fly—t is initialised to 0 and is being incremented by 1 every time a test fails. Since $t \leq \frac{n}{l}$, where l is the number of layers, the complexity of the whole procedure is linear.

On the other end of the spectrum we have an interesting result due to LaPaugh [LaP80a, Section 4.3.1] concerning routing two-point nets across a channel where the orderings of the terminals on top and on bottom is not necessarily the same. Then the problem of minimizing the channel's width in the directional model using only one jog track per signal net is NP-complete. Since the length of the ith wire is $|a_i - b_i| + w$ (where w is the channel's width) when the one-jog strategy is used, the wire length minimization problem (total or longest) in this setting is NP-complete as well. Notice, that since no wire extends beyond the extent of its terminals, the length of the channel is simply the distance between the extreme terminals.

Szymanski has recently shown [Sz81] that minimizing the width for two-layer dogleg routing (*i.e.* arbitrary jogging is allowed) is NP-complete. His construction needs, however, four-point nets (with terminals on both sides of the channel) to support the NP-completeness result. It remains open whether minimizing width for two-layer, dogleg routing of three- or two-point nets is NP-complete as well, or is the problem tractable after all (very much to everyone's surprise)?

Channel routing in the knock-knee model is considerably more tractable: Rivest, Baratz and Miller [RBM81] show how to achieve channel width that is only a constant times the optimal solution. Their wiring strategy, however, requires routing in columns beyond the extent of the cahnnel, thus the channel's length may not be minimized together with its width.

Another interesting subject concerning channel width is establishing *lower bounds*. An obvious requirement on the width of the channel is that on every line crossing the channel there will be enough room for the wires realising the nets that have to cross that line. The *density* argument has traditionally been applied to vertical crossing lines, obtaining the following lower bound on channel width:

$$d = \max_x |\{ N_i \mid (a_i - x)(b_i - x) < 0 \}|$$

(it suffices to check one value of x between every two terminal positions). In Chapter III we saw how non-vertical crossing lines can be used to derive lower bounds in a river routing situation, but this technique does not generalize well when terminals are in different order on the bottom and the top sides of the channel.

Recently, other lower bound techniques have been developed, most notably by Brown and Rivest [BrRi81] for the directional model and by Leighton [Lei81] for the knock-knee model. The techniques of [BrRi81] draw heavilly on the fact that no two wires can share a corner. If the number of nets for which $a_i \neq b_i$ is m, and the horizontal distance between the rightmost and the leftmost terminals is e, then the number of tracks required is $t \geq -(e-n) + \lceil \sqrt{(e-n)^2 + 2m} \rceil$, allowing wires to be routed beyond the extent of the channel. In fact, their argument can be used to show that if routing is allowed only within the extent of the terminals (*i.e.* using $e+1$ columns), then $t = \Omega(m/e - m)$*. In [Lei81] a channel routing problem is described for which the smallest attainable width is $t = 2d - 1$ if the knock-knee model is being used. This matches the performance achieved by the algorithms in [RBM81], thus existentially the lower and upper bounds agree.

The major question that remains open concerning lower bounds is whether the relative orderings between the terminals can be incorporated into the derivation. In other words, can the structure of the permutation (on nets) being realized by the channel be used to obtain tighter

* For two functions f, g we say that $f = \Omega(g)$ iff $g = O(f)$.

bounds? From previous work we have learned that the relative *positioning* of the terminals is sometimes more important than their ordering (e.g. river routing), but the ordering—no doubt—plays an important role by itself.

2.2. Minimising Layers and Width in the Via-free Model

In this section we first show how to efficiently determine the minimum number of layers required to route two-point nets across a channel using the via-free wiring model. Then we show how to find the minimum width of a channel when the smallest number of layers is being used in the via-free wiring model. This is done by combining the framework developed for the layer minimization problem with our earlier work on river-routing across a channel (Chapter III).

In the via-free model, each net is being assigned exactly one layer in which the net is to be realized. Two nets have to be assigned different layers if and only if they *cross* each other, that is if the straight lines drawn to connect their terminals intersect. In symbols, the crossing condition for nets N_i and N_j is $(a_i - a_j)(b_i - b_j) < 0$.

A set of nets going across a channel can be assigned l layers so that no two crossing nets will be assigned the same layer if and only if the following coloring problem can be solved: Given are l colors and the graph $G = (V, E)$ where $V = \{N_i,\ i = 1, \ldots n\}$ is the set of nets, and $E = \{(N_i, N_j) \mid N_i \text{ crosses } N_j\}$ represents crossings. The coloring problem in general is NP-complete, but the kind of graphs set up by the channel routing problem in the via-free model is a special class called *permutation graphs*. Such graphs can be colored in polynomial time for any number of layers l. In case, however, *single-sided* connections are allowed, the channel routing problem becomes NP-complete since it is equivalent to the problem of coloring a *circle graph* [GaJo79].

Even, Pnueli, and Lempel [EPL72] have provided an $O(n^2)$ algorithm for coloring a permutaion graph with n nodes. In fact, they recognized the applicability of such graphs to the very same problem discussed here (cast in a PCB context). Here we shall describe how their algorithm can be modified to run in time $O(n \log l)$, where n is the number of nets and l is the number of layers.

To be consistent with [EPL72], let us number the nets so that the terminals on the top side of the channel read 1 to n when going from left to right (*i.e.* $a_1 < a_2 < \ldots < a_n$). The net names of the bottom terminals, when read from left to right, form a permutaion of the numbers 1 to n, which is the permuation of the graph (or the permutation realized by the channel). We also associate with each bottom terminal its ordinal as obtained when scanning the bottom side, *i.e.* we associate the number 1 with the leftmost bottom terminal, 2 with the second from the left, etc. p_i denotes the ordinal associated with net N_i. An example of the canonical net numbering

and the associated p_i's is given in Figure 2: $p_1 = 4$ since b_1 is the fourth terminal from the left on the bottom side.

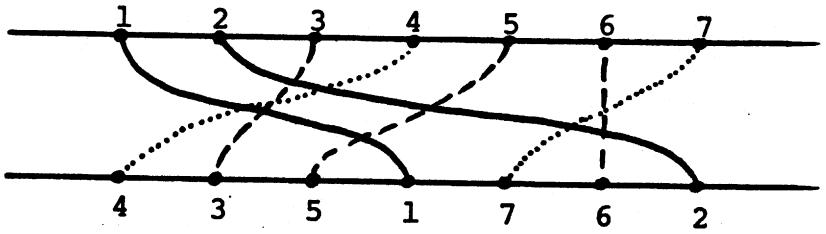


Figure V.2: Minimal layer assignment in the via-free model.

We denote the layer assigned to net N_i by $L(i)$, and for each layer k we define $p_{\max}(k) = \max\{ p_i \mid L(i) = k \}$. Nets are routed in increasing order by net number, and layers are assigned according to the following rule: $L(i) \leftarrow k$ such that $p_{\max}(k) < b_i$ is largest. In other words, for each layer we look at the rightmost net assigned to it; we eliminate all rightmost nets that cross N_i, and then choose the layer whose bottom terminal is the closest to b_i. In the example of Figure 2, nets 1 and 2 are routed in the same layer, and then 3 and 4 are routed in a second and third layer. Net 5 can be routed either in $L(3)$ or $L(4)$, but since $p_{\max}(L(4)) = 1$ and $p_{\max}(L(3)) = 2$, we set $L(5) \leftarrow L(3)$. Net 6 is then routed in $L(5)$, and finally $L(7) \leftarrow L(4)$ following the same rule.

Notice that we have been intentionally vague about the *set* of layers over which we iterate. This is because the same algorithm, essentially, can be used both for the minimization and the decision version of the problem, *i.e.* we can either ask "how many layers are required to realize the interconnect pattern?" or ask "are l layers enough?" and use the same algorithm to answer them. In the minimization version we try to use existing layers as long as possible; only when a net is encountered that has to be on a new layer do we use up a new layer. On the other hand, when the number of layers is given we could either adhere to the same philosophy and claim "failure" when we run out of layers, or—alternately—we can initialize $p_{\max}(k) \leftarrow -\infty$ for all layers $k = 1, \ldots, l$ and run the algorithm until a net for which a layer cannot be assigned is found.

The correctness of the procedure devised in [EPL72] applies to our algorithm as well. Their algorithm is slightly different in the rule for selecting the next layer: rather than looking for the closest rightmost net on the bottom, they look for the farthest rightmost net on the top. Formally, their algorithm can be reformulated* by first defining $q_{\max}(k) = \max\{ i \mid L(i) = k \}$, and then the layer selection rule is: $l(i) \leftarrow k$ such that $j = q_{\max}(k)$ is the smallest for which $p_j < p_i$, *i.e.* N_i and N_j are non-crossing. The argument in the correctness proof in [EPL72] can be now repeated verbatim for our selection rule.

* [EPL72] does not include such specific instructions, it is only the proof that implies their necessity.

The advantage of looking for the closest bottom terminal rather than the farthest top terminal is in improving the worst-case running time of the algorithm from $O(n^2)$ to $O(n \log l)$. We conduct a binary search on the $p_{\max}(k)$'s using p_i as a key; as soon as we find the largest $p_{\max}(k)$ that is smaller then p_i we can assign $L(i) \leftarrow k$. Each such search takes time at most $O(\log l)$ since the test for non-crossing between two nets is a constant time operation.

Now we turn to the subject of minimizing the channel's width in the via-free model. The interaction between wires as far as spacing is concerned occurs in each layer independently of the other layers. Thus we can partition the wires into set according to the layer they are routed in. We can exploit the river routing constraints as developed in Chapter III in order to determine the width that is required for routability: whenever a layer is being assigned to a net, we test whether it will fit in the specified (or current, depending whether we are solving the decision or the minimization problem) width.

Unfortunately, in order to obtain the minimum width attainable with the minimum number of layers required, we have to resort to the layer assignment rule from [EPL72], rather than the new rule devised here. The reason is that choosing the farhtest—rather than the closest—net guaranteed maximal usage of the available space. In the example of Figure 3 we see how the two layer assignment procedures differ. Although the number of layers utilized is the same (2), the selection in 3(a)—following [EPL72]—yields a narrower channel than the selection in 3(b).

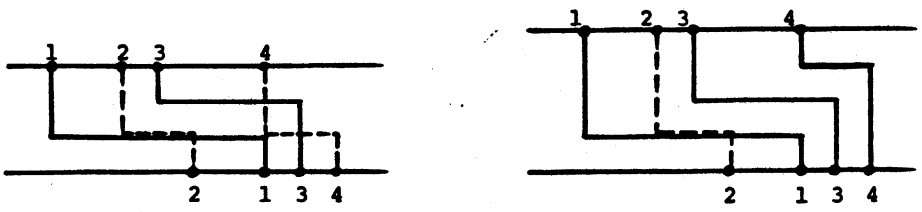


Figure V.3: Two solutions to the same channel routing problem using two layers in the via-free model. Each has minimum width relative to its layer assignment, but (a) is narrower than (b).

Another point concerens the relationship between the minimum width and the minimum number of layers. The width minimization works only when the routing is done with the minimum number of layers needed. If more layers are available, the extra degree of freedom makes the optimization problem too compicated to handle with the same technique. We tried to use dynamic programming to solve the general decision problem, in which l and w are given and the question is "can the channel be routed in l layers within w tracks?", but we were not able to devise a proper criterion. The only situation for which this problem is solved is river routing (see Section III.5).

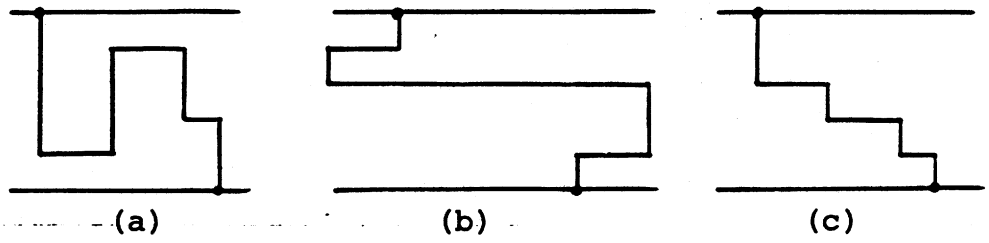


Figure V.4: Monotonic routing across a channel: (a) is monotonic only in the horizontal sense, (b) is monotonic only in the vertical sense, and (c) is monotonic in both directions.

3. Monotonicity

A wire going across a channel is routed in *horizontally monotonic* fashion if the x-coordinates of its jog columns form a monotone sequence (see Figure 4(a)). Similarly, the y-coordinates of the jog tracks of a *vertically monotonic* wire must form a monotonically decreasing sequence (see Figure 4(b)). Intuitively, a wire is horizontally monotonic if when traversing its path from one terminal, all turns from vertical to horizontal segments are made towards the other terminal, and similarly for vertical monotonicity. We say that a wire is *monotonic*) if it is monotonic *both* horizontally and vertically (Figure 4(c)).

It has been shown recently ([BrRi81] and [LloRa81]) for the directional two-layer model that using horizontally nonmonotonic routing can reduce a channel's width by a factor as large as $\Theta(\sqrt{n})$* in comparison with the optimal solution achievable with horizontal monotonic routing. The savings in width is made at the expense of the channel's length. For a discussion of the effect on the area see [LloRa81]. The situation for vertical monotonicity is different.

Theorem V.1. Whenever routing two-point nets across a channel using the knock-knee wiring model, a minimum width solution can be obtained by using only vertically monotonic wires. If the directional two-layer model is being used, a solution whose width is at most 1 more than the possible minimum width can be attained in a similar manner.

(Note that a similar result for the one-layer model is trivial.)

Proof. The proof is constructive in that we show how to transform every vertically non-monotonic routing to a monotonic one without increasing the number of tracks at all for the knock-knee model, or by increasing it by at most 1 for the directional model. We start by dealing with both models together; from a certain point, the knock-knee property enables us to complete

* For two functions f, g we say that $f = \Theta(g)$ iff $f = O(g)$ and $g = O(f)$.

the construction without changing the original width of the channel. The directional case is harder and may necessitate the usage of an extra track, all in all yielding the desired result.

A *valley* is a jog track whose attached jog columns both lie above it, as in Figure 5. The horizontal tracks are scanned from bottom to top, left to right in each track. Whenever a valley is encountered (whether original or produced at an earlier stage of the procedure), we try to *fill* it. Figure 5 demonstrates this construction. We run a wire in the horizontal track that is the lower among the top ends of the jog columns adjacent to the valley and eliminate the unncecessary portions of the original wire.

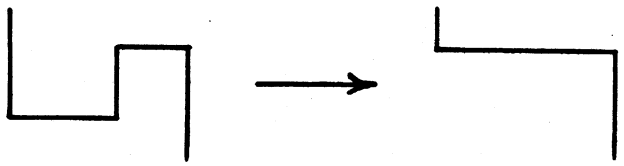


Figure V.5: Filling a valley.

As a result of filling a valley, the new wire may overlap with some existing horizontal segments in the filled area. We pull down all wires with such segments to the position freed by the valley. This may cause two kinds of problems: overlaps with other horizontal segments outside the valley, and overlaps of vertical segments anywhere in the channel. The difference in treatment between the knock-knee and the directional models depends on whether vertical overlaps occur within the valley's span (*i.e.* between its right and left ends) or not.

For the knock-knee case we abandon the filling plan, and just look for the lowest horizontal track above the valley with which the valley can be exchanged without causing any vertical or horizontal overlaps within its span. Because a valley is defined as being delimited by two upwards turning jog columns such a track always exists strictly above the valley. Notice that by this operation we have raised the floor of the valley to a higher track than before if not eliminated it altogether. At this point only conflicts outside the valley remain. In the knock-knee model they can be virtually ignored: we can use the geometry of wires as it was before the valley was processed and switch the assignment of wires to nets without changing the geometry. Necessary connections can be made by using the columns freed by the vertical segments adjacent to the original valley (as can be seen in the transition from Figure 6(a) to 6(b)).

The directional case is much harder, and we just mention briefly the main points: here we insist on "filling" the valley the first time we encounter it. This is done by forming a *chain of interferences* due to overlaps of both kinds — within and outside the valley's span: instead of pulling the wires from the fill-track all the way down to the valley's track, we pull them only as far down as possible in compliance with the interferences, thereby forcing another set of segments down to the track with which they interfere etc. If we are lucky, we shall be able to accomodate

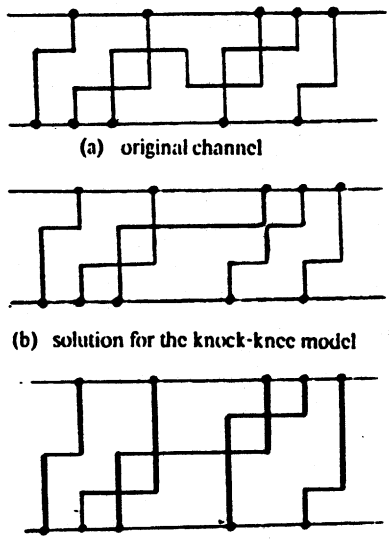


Figure V.6: Converting a vertically non-monotonic route to a monotonic one. The original channel is shown in (a), solution for the knock-knee model in (b), and the solution for the directional model in (c). In (c), an extra track is needed.

the whole chain by the time we reach the valley; otherwise, we need one extra track. In the latter case, we take the wires that were originally routed in the fill-track and put them in a specially created track at the top of the channel, and leave the rest of the chain intact. This is partially demonstrated in the transition from Figure 6(a) to 6(c).

In both cases the transformation may create new valleys. However, as long as the tracks on which valleys are sought are scanned from the bottom upwards, and the leftmost valley of the lowest track is processed first, we shall eventually eliminate all valleys. Note also that the extra track introduced in the directional case can be shared between all valleys that need it. Proving the convergence of this process is lengthy and relies heavilly on properties of the models and the definition of a valley. ∎

Remark. The procedure used in the above proof outline is not tight in the sense that it may create an extra horizontal track even when such a track is not required. This may happen when a minimum width routing can be obtained by "straightening out" a valley and its adjacent plateau, rather than filling the valley. We could not find a procedure that will take care of this phenomenon. The example of Figure 6, however, demonstrates that the additional track may be indeed required in order to achieve vertically monotonic routing.

The result of Theorem 1 assures us that if we are willing to sacrifice one horizontal track, we should not waste any resources looking at solutions involving vertical non-monotonic wires when employing a heuristic. This is in contrast with what is believed to be the case for channels including signal-nets with three terminals or more. In fact, popular heuristics for that case

([Deu76], [YoKu82]) make heavy use of such routes. Moreover, this result provides a bound on the *length* of the routing needed to be done in the layer chosen to implement vertical segments. The bound for each net is simply the width of the channel. Since in nMOS circuits the resistance of various conducting layers differs considerably, and we tend to minimize channel width, this result indicates that we should use metal for the horizontal direction and polysilicon or diffusion for the vertical one.

4. Jogging

There are various reasons why excessive *jogging, i.e.* including relatively many turning points in the wiring, should be avoided. The major concern is the size of the data structures maintained by the layout procedure because the number of turning points is an obvious lower bound on the space complexity of any algorithm generating the wiring or dealing with it afterwards. Also, limiting the amount of jogging cuts down the search space required by a heuristic procedure. Thus it would be desirable to have some kind of a guarantee that bounding the amount of jogging is not going to hurt us too much in terms of optimality.

Another aspect of the jogging problem is electrical performance. In the via free model, and sometimes in the knock-knee model, turning points are corners on a single layer. Depending on the fabrication process, the resistance per unit square in such corners may be as much as 56% higher than along straight lines [Zi81], and we would like to avoid a cumulative effect of such phenomena*. In the directional model, and—again—sometimes in the knock-knee model, the problem is even worse. Each turning point is implemented as a contact between two layers. Then not only extra resistance is being incurred, but also more area is required, forcing us to use a larger grid size.

Storer [Sto80] has studied the relationship between the area required to lay out a planar graph on a grid and the number of turns that edges make. He provides evidence to the effect that minimizing area may require arbitrarily more turns than optimal and vice versa.

In this section we shall discuss the relationship between the number of jogs in a solution and its optimality. We limit our setting to routing two-point nets across a channel, but we discuss both planar and nonplanar patterns.

One well understood routing situation is *river routing*. In Chapter III we saw how to use the greedy ("contour hugging") algorithm to obtain solutions that are optimal both with respect to the width and the length of the channel. Unfortunately, there are river-routing situations that require a large number of jogs so that full advantage can be taken of a channel's capacity when it

* Sometimes, however, the *interaction* between such phenomena alleviates their individual effects, and the end result is not additive. The analysis in this case is not just quantitative—it requires a closer examination of the geometry of wires. Also, better circuit level models are needed, and this is beyond the scope of this thesis.

is low. To route n signal nets, we may need as many as $\Omega(n^2)$ turning points; this is exemplified by Figure 8(a). On the other hand, the greedy routing algorithm may generate more jogs than are necessary in less constrained situations, as shown in Figure 8(b). We now give an algorithm that is essentially as fast as the greedy algorithm to river route a channel using the minimum possible number of jogs.

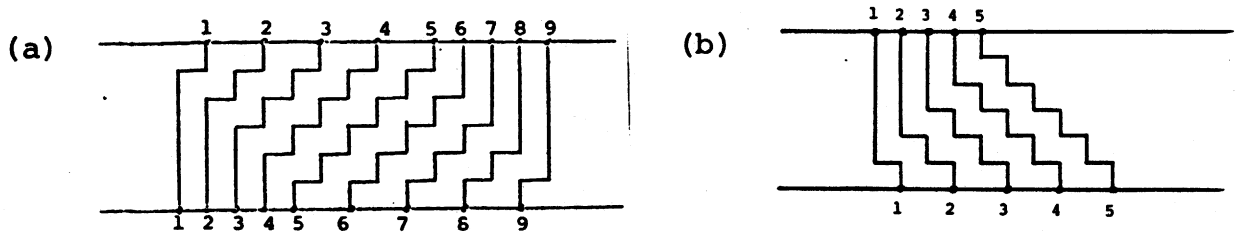


Figure V.8: River-routing: (a) shows a situation requiring $\Omega(n^2)$ jogs, whereas in (b) more jogs than necessary are generated by the greedy algorithm.

Similarly to the greedy algorithm, this procedure routes the channel net by net, scanning it once from left to right, in the order that the nets are numbered. Nets that are pitch-aligned ($a_i = b_i$) are routed straight across. The treatment of *rising* nets ($a_i > b_i$) is symmetric to that of *falling* nets ($a_i < b_i$), and for sake of brevity we shall discuss the latter case alone.

Here is a semiformal specification of the routing procedure for one net (in pseudo-Algol):

Algorithm V.1. Routing a falling net so as to minimize jogs

route vertically down from a_i as far as possible;
$t \leftarrow$ number of tracks crossed by above segment;
<u>while</u> $a_{i+t} - b_i < t$ <u>do</u>
  jog once along contour;
  $t \rightarrow t +$length of last vertical segment
  <u>od</u>;
route horizontally to b_i's column and down to b_i;

All we have done was to modify the greedy algorithm by adding the following rule. Whenever a net is about to take a downward turn, we check whether routing it straight horizontally all the way to its target column will not render the nets on its right unroutable. This test is simple and quick (involves one comparison) using the formulation cast in Section III.1 (Condition (2)). We apply the test to a contracted channel consisting only of the tracks that have already been crossed by the wire, including the one reached so far. The relevant number of tracks is accumulated in the variable t. If the test succeeds, no jogging is required, and thus we simply route the wire horizontally to its target location. Otherwise, we turn downwards right away (at the next

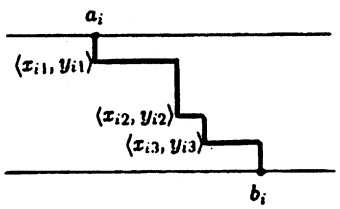


Figure V.9: Representation of the ith wire.

available column, *i.e.* follow the contour for one turn) and route as far as possible towards the other side. Then we start a new horizontal segment and apply the test again when a possibility to turn downwards arises, and so on until we reach the destination.

In order to implement any algorithm of this kind, including the greedy algorithm, one needs a data-structure for wires and *contours*. Our suggestion is the following. Since all wires are monotonic and rectilinear, it suffices to store every other corner, starting from the first jog. For falling nets this means that we represent a wire by its bottom-left corners as in Figure 9, and we also include the terminal positions in the description. Thus a falling wire i is represented by the list $(a_i, \langle x_{i1}, y_{i1}\rangle, \ldots, \langle x_{il_i}, y_{il_i}\rangle, b_i)$, where $a_i = x_{i1}$, the x–coordinates form an increasing series, and the y–coordinates form a decreasing series. A rising wire is described by upper-left corners, and the only difference in the representation is that both the x-coordinates and the y-coordinates are monotonically increasing. Notice that a description of the ith wire provides a description of the contour needed for routing the $(i+1)$st wire.

Evidently, the time required to run this algorithm is bounded by a constant times the number of jogs (or wire segments) it produces, thus it is optimal in the complexity sense. Moreover, it indeed generates the minimum number of jogs for the channel and thus is optimal in the quality of its results as well. It turns out that minimising the total number of jogs also attains the minimum number of jogs for each net individually. Trying to minimize some other function of the distribution of the number of jogs (such as standard deviation) seems to be a much harder problem, though. We summarize by stating:

Theorem V.2. For a channel with a given width, Algorithm 1 (when applied to all nets) river-routes it using the minimum number of jogs per net (and subsequently, the total number as well). The algorithm's time and space complexities are linear in the size of its output.

Proof. We distinguish between rising *blocks* and falling ones (following left-going and right-going blocks in [DKSSU81]). A rising block is a maximal set of consecutive rising nets, and likewise for a falling block. The channel can be partitioned into such adjacent blocks because the routing within one block do not interfere with what happens in the other blocks and hence it is enough to prove the theorem for one block. Without loss of generality we choose a falling block.

There are two key observations. First, jogs are taken only if they are necessary to enable routability of the remaining nets. Second, if a jog has to be taken it had better occur as soon as possible (*i.e.* along the contour). We must show that each jog generated by this algorithm on any wire is mandated by the channel's configuration. This is done by induction on nets: net i attains the minimum number of jogs it requires to ensure routability provided that nets 1 through $i-1$ were routed using the algorithm. The basis, for net 1, is obvious. The induction step relies on the surprising fact that once a contour is being followed for one turn—it has to be followed all the way except for, possibly, the very last turn. (Algorithm 1 can be subsequently simplified to save unnecessary comparisons.) This ensures that no extra jogging in wire i can save some subsequent jogging in net $j > i$. In other words, the greedy approach to jogging in necessitated by the routing constraints.

Next we show that Algorithm 1 (as stated) indeed produces jogging only in the above pattern.

∎

Now recall LaPaugh's NP-completeness result concerning minimizing width in the directional model that requires one jog per net [LaP80a, Section 4.3.1]. In the situation of routing two point nets across a channel when the orderings among the terminals on top and on bottom are unrelated, the problem of minimizing the channel's width in the directional model using only one jog track per signal net is NP-complete. This result has special meaning in the context of this section because it tells us that even if we limit ourselves considerably in terms of how much jogging we allow, we are not likely (unless P=NP) to find a polynomial-time algorithm for routing which will be optimal with respect to to the width criterion. Notice, of course, that since only one horizontal segment is allowed for each net, the length of the channel is simply the distance between the extreme terminals.

In this light, the following result is interesting:

Theorem V.3. For both the directional and the knock-knee wiring models, there exist routing configurations for n two-point nets across a channel in which any minimum width monotonic routing will force one of the wires to have $\Omega(\sqrt{n})$ jog tracks.

Proof. The constructions given in Figure 10 provide the pathological cases. For simplicity, we used definite configurations, but they can be understood as generic patterns. In the directional case of Figure 10(a), each configuration is made up of *blocks* of signal nets labeled contiguously

from some index l to some index r, such that $b_i = a_i + 1$ for $i = l, \ldots, r$. In the knock-knee case of Figure 10(b) the structure of the block is slightly more complicated, as shown. If k is the number of jogs we want to force, we use $k-1$ blocks of $k-2$ nets each, plus 2 blocks with $k-1$ nets per block. Adding on the forced net (having the largest label), we have all in all $k \cdot (k-1)+1$ nets, achieving the desired ratio between jogs and number of nets. In Figure 10, we chose $k = 4$.

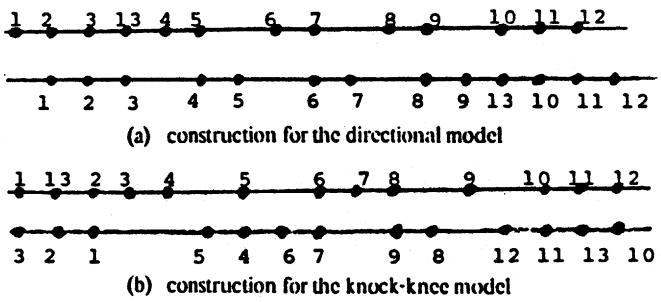
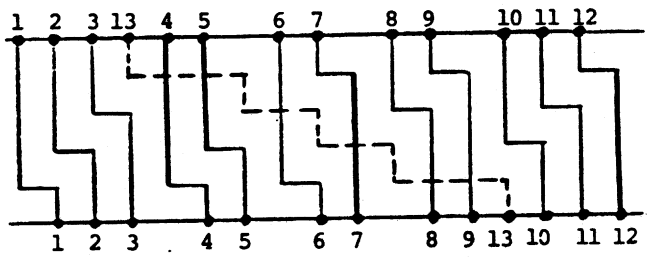
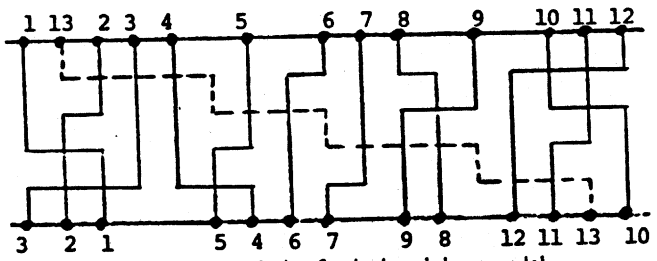


Figure V.10: Channels forcing net 13 to jog 4 times. (a) is the construction for the directional model, and (b) is the construction for the knock-knee model.

In Figure 11, we see how a width of $k+1$ (*i.e.* routing in k tracks) can be obtained. This is optimal due to a density arguments. It remains to be shown that in every optimal routing the forced net is required to have k jog tracks. We prove this by induction on k. At each step we splice the additional nets into the existing structure, showing that anything short of jogging through the blocks will make a monotonic routing of the forced net impossible. ∎



(a) optimal solution for the directional model



(b) optimal solution for the knock-knee model

Figure V.11: Solutions for the channels of Figure 10. (a) and (b) give the optimal solutions for the directional and knock-knee models, respectively.

Remark. Notice that here, unlike the situation of Theorem 1, relaxing the minimum width goal by allowing it to be 1 or any other constant more than optimum does not help — the lower bound on the number of jogs is still essentially the same.

To summarize, we have seen that on one hand, jogging is essential in obtaining minimum width channels, but on the other hand limiting its extent does not make the problem of finding optimal solutions in general any easier.

5. Resolution of Conflict Cycles

A major problem in channel routing is that of *vertical conflicts*. In all models which disallow overlaps of wires and corners, if two terminals with different labels appear in the same column (*i.e.* have the same x-coordinate) we must assign a higher jog track (in this column) to the top terminal. This may impair the quality of the routing, and sometimes even make it impossible.

The notion of a *vertical conflict graph*, due to Hashimoto and Stevens [HaSt71], has been widely used to model this problem. We associate one vertex with each signal net, and draw a directed edge from one vertex to another if a terminal of the first occurs under a terminal of the second in some column. Figure 12 shows a channel and its corresponding graph. It is easy to see that if we constrain ourselves to using only one jog track per net, no routing can be found whenever the vertical conflict graph contains a directed cycle. On the other hand, we show in

this section that resolution of cycles can be obtained using (almost) monotonic routing except for extreme cases and only with a slight deviation from the one jog track per signal net restriction.

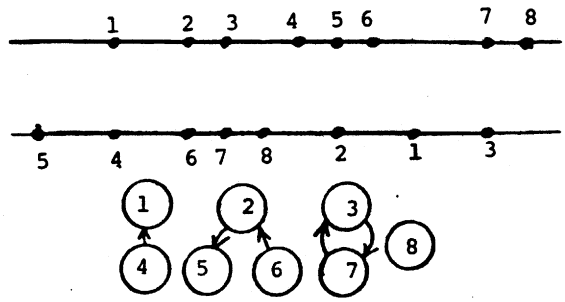


Figure V.12: A channel and its corresponding vertical conflict graph.

The vertical conflict graph is well defined for channels whose nets have more than two terminals, but its usefulness decreases as the size of the nets grows. This is demonstrated in [YoKu82], where the nodes of the vertical conflict graph are split (according to the positions of internal terminals) in order to achieve smaller channel width. From here on, we limit the discussion again to two-point nets across a channel and to monotonic routings.

First, we consider nets whose terminals occur in the same column. The only way they can be routed monotonically is straight across, thus taking up all the routing space in their column and not interacting with the rest of the channel. Hence we eliminate these nets altogether from the channel by removing their column and contracting the channel accordingly. This simplification concerns both the actual routing and the definition to follow, making it simpler to state.

Definition V.1. A cycle in the vertical conflict graph is *tight* if the nets involved in it occupy a contiguous block of columns. A cycle that is not tight is called *loose*.

The next two lemmas establish the relationship between tightness and routability.

Lemma V.1. A set of nets corresponding to a tight conflict cycle cannot be routed monotonically.

Proof. Since there are only two terminals per net, the number of columns in the block must be equal to the number of nets in the cycle. The proof proceeds by induction on the number of nets. For $n = 2$ we have the infamous X crossover that is unroutable by inspection (Figure 13). If a tight cycle of $n = k$ nets cannot be routed, neither can a tight cycle of $n = k + 1$ nets, because the new net essentially establishes an X withthe rest of the nets. ∎

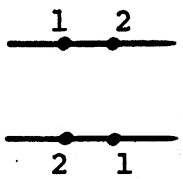


Figure V.13: An X crossover: no horizontally monotonic routing exists.

Lemma V.2. A channel consisting solely of nets corresponding to one loose conflict cycle can be routed monotonically so that each net but one uses only one jog track; the remaining net uses two jog tracks.

Proof. The order in which nets are routed is determined by the conflict cycle: starting with any net, we route each net using one horizontal track and then route the next net on the cycle. This can be done successfully until we try to route the last net, which has to broken into two tracks. Since the cycle is not tight, there exists one column in the channel that can be used to switch from one track to another. To ensure (horizontal) monotonicity, we make sure that the last net to be routed (which precedes the first net in the cycle) has one terminal to the left of the free column and the other one to its right. One such net always exists by the definition of a loose cycle. The result of this routing strategy is demonstrated in Figure 14, where we started routing from net 4, ending up by splitting net 1 between two tracks. ∎

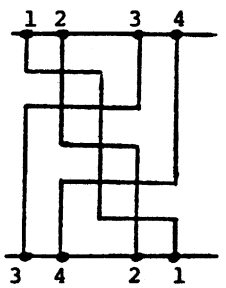


Figure V.14: Resolution of a single vertical conflict cycle.

The more interesting case is when cycles interact:

Definition V.2. A set of nets corresponding to a union of disjoint cycles in the conflict graph is called *tight* if their terminals occupy a contiguous set of columns. A set of nets that is not tight is called *loose*. The *span* of a set of nets is the set of columns enclosed between the leftmost and rightmost columns belonging to any of the nets.

A tight set of nets corresponding to three cycles is shown in Figure 15. A set of nets is loose if there is even one column in its span that has a terminal of a net not involved in any of the cycles.

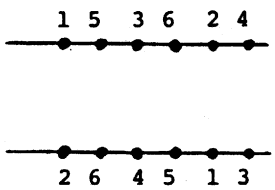


Figure V.15: Three cycles forming a tight set of nets.

The following result is an extension of Lemma 1.

Lemma V.3. A tight set of nets cannot be routed monotonically.

In order to obtain the main result of this section, we have to relax our monotonicity requirement slightly.

Theorem V.4. Any loose set of nets can be routed within the span of the set so that each net but one per cycle uses only one jog track. The remaining nets use two jog tracks each.

Proof. Clearly, it is enough to look at a set of nets corresponding to a union of cycles. Also, if we can show that a loose set containing only one extra column can be routed in the fashion described in the Theorem, we are done. The proof is by induction on the number of cycles: we can use the same free column to resolve all cycles (as in Lemma 2). This may force the routing of one net per cycle to be horizontally nonmonotonic, but the routing is still within the span of the original set. ∎

If we use the routing scheme devised in the proof, e end up having one column that has as many vertical jogs as the number of cycles. This may be cumbersome when it comes to the representation of the routing in a data structure, so a more elegant construction can be used that splices the cycles into one another as shown in Figure 16. The resulting routing contains in each column at most one vertical segment connected to a top terminal, one connected to a bottom terminal and maybe one dangling in between (the number of such dangling segments is equal to the number of cycles).

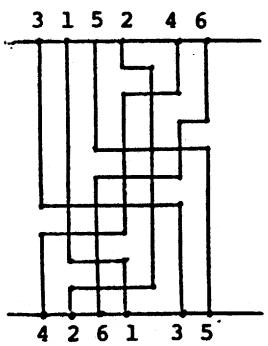


Figure V.16: Resolution of a non-tight set of nets involving three cycles.

Professor Jeffrey D. Ullman from Stanford University has pointed my attention to the fact that a similar idea was applied in [KaKa79] to expandable channels. The setting in their paper is rather different from ours, but it is interesting to note the versatility of this technique.

This routing strategy has an interesting application to *restructurable* VLSI [Ra79], as has been brought to my attention by Sandeep Bhatt. In this technology, metal connections can be made or broken after fabrication by laser welding. Now note that a permutation of n elements can have at most $\frac{n}{2}$ cycles. Using our technique, we can realize any such permutation by a crossbar switch layout of dimensions $(n+1) \times \frac{3}{2}n$, rather than the obvious $n \times 2n$ layout.

All in all, we have shown how cycles can be resolved using a simple strategy in most cases. We have, however, discussed only the issue of routability, not the one of optimality. If there is only one free column in a loose set of nets (as in Figure 16), the solution provided is indeed optimal for width, but the problem of relating the width to the number of free columns remains open.

6. Routing Across a Rectangle in Arbitrary Order

Consider the case in which a side of the channel belongs to a module in which the order of the signals can be changed arbitrarily by easy changes to the module, e.g. I/O to a PLA, I/O pads, or data registers. Then a set of terminals is associated with a set of signals of equal cardinality, and we create the individual association between signals and terminals as part of the routing procedure. In this case we might take advantage of the freedom we are given and reduce the width of the channel or the number of layers that are needed by using an optimal algorithm for a certain type of ordering, such as river routing. Here we shall discuss the problem in the context of the directional model, for which an optimal algorithm exists if terminals are allowed to be connected in arbitrary order. The savings in width earned by such an algorithm is exemplified by Figure 17(a).

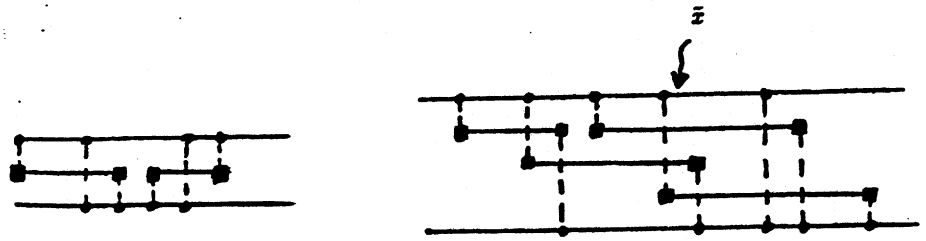


Figure V.17: Routing across a channel with arbitrary terminal ordering (notice the terminals have no labels). (a) shows a simple case using only one horizontal track; (b) demonstrates a more complicated case with $e_C = 3$ (due to $\bar{x}$).

The key observation is that terminals which are aligned across the channel must be connected straight across. The rest are processed in order from left to right (on the top side) and each is connected to the first available terminal on the bottom side. Each wire is assigned the highest available horizontal track as its only jog track. An example of this procedure is shown in Figure 17(b).

We restrict ourselves to a horizontal channel, C, having terminals on two opposite sides only. Let $l_T(x_0)$ and $l_B(x_0)$) be the number of terminals on the top and bottom side, respectively, of the channel lying to the left of x_0. Then we define the *excess number at a point* $x = x_0$ to be*

$$e_C(x_0) = |l_T(x_0) - l_B(x_0)|.$$

Equivalently, we could have defined excess number with terminals lying to the right of x_0. The *excess number of the channel*, e_C, is defined as

$$e_C = \max_{x_L \le x \le x_R} e_C(x)$$

where x_L and x_R are the x–coordinates of the left and right ends, respectively, of the channel.

Theorem V.5. The number of horizontal tracks needed to route C is exactly E_C, which is optimal.

Proof. The number of e_C tracks can be attained by first routing aligned terminals straight across and then assign horizontal jog tracks using the track assignment rule mentioned above. Note that this algorithm does not cause two vertical tracks to overlap, as opposed to a similar case in [DKSSU81] (see Figure 17(b)). Also, no wire passes through more than two contacts. The lower bound is proven by drawing vertical line segments through the channel. By the definition of e_C at least at some point x as many as $e_C(x)$ signals have to be routed from the left side of x to its right side, thus forcing us to use as many as e_C tracks. ∎

The calculation of the excess number is linear once the terminals are sorted. The assignment of tracks is also linear. The tracks can be maintained in a free-list, because it does not matter which track is assigned to which net as long as they do not overlap. Remember that there are no vertical conflicts, because all aligned terminals have been connected at the beginning. All in all we have linear time algorithms both for testing routability and for routing the channel (assuming the terminals are presorted). If they are not, the complexity is $O(n \log n)$, where n is the number of terminals on each side.

This result can be generalized to dealing with disjoint sets of signals. The order of terminals within each is arbitrary, but we are not allowed to mix terminals from different sets. This

* x_0 need not be integral; in fact, we must look at points of the form $l + \frac{1}{2}$ where l is an integer. Moreover, it is superfluous to look at integral points. It is enough to look at points just before $(-\frac{1}{2})$ and just after $(+\frac{1}{2})$ terminals for the application to follow.

is achieved by modifying the definition of the excess number to accommodate this constraint. Furthermore, the excess number has the same bitonicity property as the conflict number discussed in [DKSSU81], thus it can be used to solve the offset problem in the same fashion as described there.

7. Conclusions

A few topics concerning the routing of two-point nets have been exposed and analyzed. Many open problems remain and promise some exciting research in the field.

- Can we improve the result of Theorem 3 by finding a channel with n nets such that one net is forced to have more than $O(\sqrt{n})$ jogs? What can be said about the average number of jogs per net?
- Szymanski has recently shown ([Sz81]) that dogleg routing, allowing an arbitrary number of jog tracks per net, is NP-complete even when terminals are located only on two opposite sides and all routing is horizontally monotonic. To support his result, however, he needs four-point nets. Can this be extended to two- or three-point nets?
- Can we derive lower bounds for channel width that will incorporate the relative orderings between the terminals? A first success in showing that density is not a tight lower bound is [BrRi81]. Can we improve their result?
- How do conflict cycles affect the optimality of channel width?
- [RBM81] includes some limited exploitation of the general two-layer model. What can be gained by using all the freedom allowed by this model? On the other hand, can we find non-trivial lower bounds for it? Notice that now the density is half of its traditional value, since wires can share tracks.

Two-point nets going across a channel can be used in the decomposition of a general channel problem into a collection of simpler ones. This can be done either by conceptually breaking up more complicated nets into related parts having two terminals each (such as in [YoKu82]), or by handling two-point nets separately ([Bar81] for example). Moreover, some of the lessons learned from the study of two-point nets can be applied successfully to nets with more than two terminals, but intricate interactions between such nets are very subtle and call for a concentrated research effort.

Chapter VI

Routing In and Around Junctions

The decomposition of the routing area available on a chip into rectangular channels is somewhat orthogonal to the global nature of the routing problem. The way that one signal net is routed affects the potential solutions for other nets. This global characteristic is one of the major reasons why routing is such a hard problem. In channel routing, the global nature of the routing patterns is fragmented by the channel structure in a way that makes a solution even harder to achieve. As a result of this approach placement evaluation becomes quite inaccurate as well.

In this chapter, we propose to look at routing problems in nonrectangular channels while still maintaining rectilinear sides. As long as modules are rectangular, such channels take one of three general shapes: T, X or L, as indicated in Figure 1. While the L's are relatively easy to handle, the other two are more complicated. Some instances of T's and X's yield to some interesting theoretical analyses which are presented in this chapter. In general, non-rectangular channels are treated by partitioning them internally along edges and dealing with each section separately. The edges are used to maintain *constraints* in such a way that overall optimality is achieved. We develop a powerful algebraic abstraction for constraint propagation, called *pairwise ordering*, which is well suited to the problem, and study it carefully.

The theoretical research on routing in rectangles has thus far paid little attention to configurations which are common in practice, and even less to the problem of propagating routing constraints through the channel. In order to fill the gap, we should examine routing of useful patterns both in T and X shaped channels as well as in rectangular ones.

*** Talk about orderings here ***

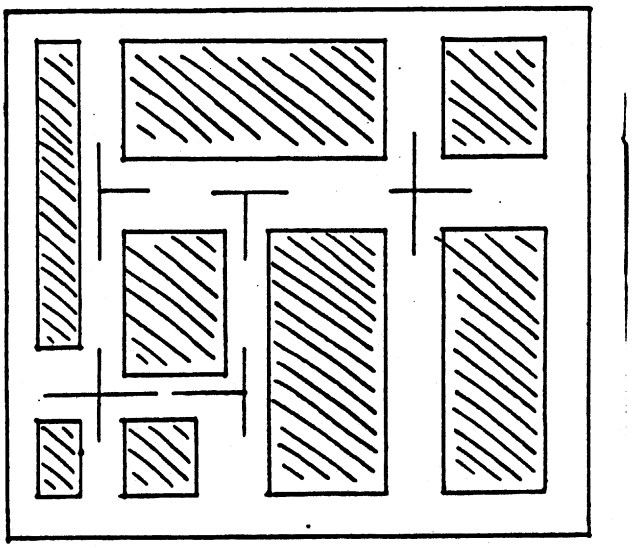


Figure VI.1: T and X junctions appear frequently in a typical layout.

The discussion in this chapter is limited to the Manhattan and knock-knee two-layer wiring models. The first two sections of this chapter describe polynomial-time algorithms for attaining optimal routings for certain configurations in T-shaped channel (Section 1) and X-shaped channels (Section 2). The notion of pairwise ordering is defined and discussed in the beginning of Section 2. Then we recall our results on routing between arbitrarily ordered terminals lying along a channel's side (from Section V.6), and conclude with a discussion of implications on a methodology for generalized channel routing.

1. Routing in T-shaped Channels

A T-shaped channel is shown in Figure 2. Its sides are named *top, flanks, legs* and *ends.* These names are qualified by the appropriate directions when needed. In general, terminals can lie on any side and *routing the channel* means to connect terminals with the same labels to each other using paths lying within the channel, obeying the design rules.

The decision problem associated with routing a channel is: "Can we route the channel in its given dimensions?" A procedure for solving this problem can be relevant in the placement phase of layout. The related minimization problem is somewhat more interesting: "How can we *minimize* the area required in order to route the channel?" This gives rise to an ambiguity because even if we assume that the sides of the channel (except for the ends) belong to three modules in the natural way, as implied by Figure 2, what movements of the modules are allowed? When does the channel cease to have a T-shape? First, we decide that the flanks of the channel will remain aligned (*i.e.* share the same grid-line). Second, it seems unnatural to use the absolute area as the optimality criterion for various reasons (see [Pr79]).Thus it is natural to consider the

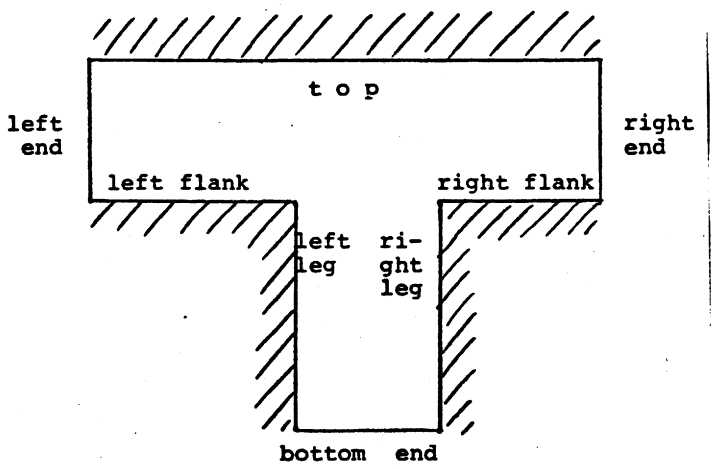


Figure VI.2: A T-shaped channel.

distance between the legs (denoted w_B, for bottom width) and the distance between the top and the flanks (denoted w_T, for top width) as our criteria in a way to be described in the following paragraph.

Moreover, it is obvious that changing the distance between the lower modules (*i.e.* changing the leg to leg distance) may effect the routing of signals going to the top and the flanks. Thus our strategy will be to minimize w_B first, and then minimize w_T with respect to it. This approach is most practical in most design situations and is also likely to approximately minimize other interesting cost functions, such as area. Notice that minimizing w_T first (by setting it to 0) will flatten out the T, *i.e.* make it into a rectangular channel by pushing the lower modules outward to the ends of the upper one. Also, once w_B is known, we can fix the horizontal location of the lower modules with respect to the top one, forming a solid T. Finally, we shall see that w_T tends to be much smaller than w_B; thus minimizing w_B first in an unconstrained manner is preferable from the placement procedure's point of view (since it is better in preserving the T-shape).

Now we restrict ourselves to *two-point nets*, *i.e.* to instances of the problem where each signal-net name can appear as the label of exactly two terminals. Also, for sake of simplicity, we exclude the ends as possible sides for terminals to lie on (they can be added at a later stage). Assuming no net connects two terminals lying on the same side or on two adjacent sides of the channel, which is reasonable if these are the sides of single modules, the nets can be divided into 5 cases according to which kinds of sides they connect:

(i) top to flanks

(ii) top to legs

(iii) flanks to legs (left to right and right to left only)

(iv) flank to flank

(v) leg to leg

The most interesting case to consider is Case (ii). Cases (i), (iv) and (v) are embedded in standard rectangular routing, whereas (iii) is essentially a restriction of (ii). We shall solve Case (ii) in the rest of this section.

1.1. Terminology and Notation

This subsection summarizes the terminology required for describing and discussing the routing results presented in the next subsection. We assume here that the flanks have been positioned relative to the top, which makes sense since the leg-to-leg distance will be computed immediately from the input without involving any of the notations to follow. Most of the terminology is summarized in Figure 3.

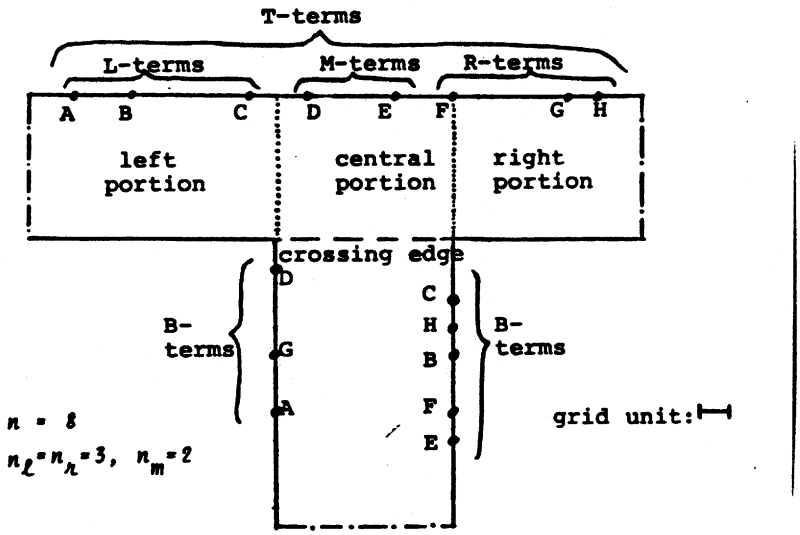


Figure VI.3: More terminology for T-shaped channels.

Here (B,G) is an aligned pair inducing a conflict; (A,F) is an aligned pair not inducing a conflict.

$n = 8$, $n_l = n_r = 3$, $n_m = 2$.

All terminals on the legs (excluding the corners) are called *B-terms*, and the terminals on the top are called *T-terms*. We denote by n the number of signal pairs; thus there are n T-terms and n B-terms. T-terms whose x-coordinate is within the range of the left (right) flank (including end points) are called *L-terms* (*R-terms*); the rest of the T-terms are called *M-terms* (for middle-terminals). We denote the number of L-terms, R-terms and M-terms by n_l, n_r and n_m, respectively (thus $n = n_l + n_r + n_m$).

Definition VI.1. Two B-terms with the same y-coordinate are called an *aligned pair*.

Alignments of B-terms induce pairing between the T-terms bearing the same labels. If the x-coordinate of the top terminal corresponding to the bottom terminal lying on the right leg is

smaller than that of the top terminal corresponding to the bottom terminal lying on the left leg, the pair is considered to be a *conflicting pair*. More precisely:

Definition VI.2. Let S_1 and S_2 be two signal nets, with corresponding top terminals S_1^T and S_2^T, respectively, and bottom terminals S_1^B and S_2^B, respectively. If S_1^B lies on the left leg, S_2^B on the right leg and they are aligned, then S_1^T and S_2^T constitute a *conflicting pair* if S_2^T lies to the left of S_1^T.

We classify conflicting pairs according to the subclasses their T-terms fall into. For all possible combinations of X, Y $\in$ {L,R,M} an XY-pair is a conflicting pair in which one T-term is an X-term and the other a Y-term*. For example the pair (B,G) in Figure 3 is an LR-pair, or equivalently an RL-pair. We order the pairs according to the positions of their T-terms, *i.e.* we write (S_1, S_2) if the x-coordinate of S_1^T is smaller than that of S_2^T. The number of XY-pairs is denoted by n_{xy}, and thus there are n_{lr} LR-pairs.

The grid-line segment going from the left end-point of the right flank to the right end-point of the left flank (dashed in Figure 3) is called the *crossing edge*. The part of the channel above it is called the *top part*, the one below it — the *bottom part*. The extension of the right leg upwards, until it hits the top (dotted in Figure 3), is called the *right edge*; the portion of the top part to its right is called the *right portion*. Likewise for the *left edge* and the *left portion*. The portion between the right and the left edges is called the *central portion*. A grid point residing on an edge and coinciding with a routing path is called a *crossing point*.

A grid-line segment enclosed by either the top part or the bottom part of the channel is called a *track*. The orientation of a track (horizontal or vertical) is relative to the T-shape, not to its parts, Tracks going from the right end to the left end, or from one leg to the other, are *horizontal* tracks. *Vertical* tracks go from the bottom end to the crossing edge, or between the flanks and the top.

1.2. Routing results

As noted above, the bottom part of the channel is routed first. By assigning one vertical track for each signal, we can easily route the B-terms to the crossing edge attaining $w_B = n+1$ by sharing horizontal tracks between aligned pairs regardless of the positions of the other B-terms (see Figure 4). The only constraint this imposes on the crossing edge is that aligned pairs will appear on it in the order corresponding to their legs. We shall see how to handle this, from the top part's point of view, whether the pair is conflicting or not. The ordering among the other signals may be arbitrary, which will be taken advantage of heavily.

* notice that there is no difference between the set of XY-pairs and the set of YX-pairs.

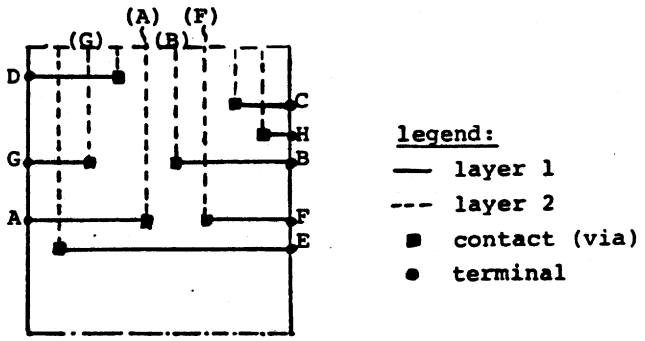


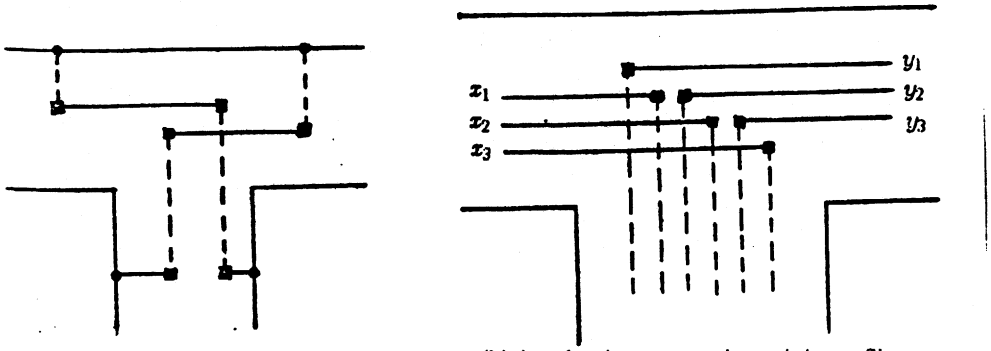
Figure VI.4: Routing the bottom part of the channel optimally and propagating its constraints to the crossing edge.

Lemma VI.1. The bottom part of a T-shaped channel can be routed optimally ($w_B = n + 1$) assuming there is no leg-to-leg routing.

Proof. The number of signals that must cross the common edge is n. Since they must abide the design rules, we need n tracks that take up width $n + 1$. ∎

This lemma is of no consequence unless we can route the top part efficiently. First we notice that signals corresponding to L-terms and R-terms can be assigned to arbitrary horizontal tracks in the left and right portions (resp.) without any loss of optimality (of w_T) there: each such (non-M-)term has to jog in order to get to the corresponding edge, so it needs a horizontal track; since there are no conflicts with respect to vertical tracks, we have complete freedom in the way we assign these horizontal tracks. Of course, some logic has to be applied in order to avoid chaos in the central portion if we want (naturally) to use the same horizontal tracks. Now, the first problem which comes to mind is how to accommodate conflicting pairs. Let us consider LR-pairs first: at first it seems that routing them will require as many as $2n_{rl}$ horizontal tracks in the central portion as might be implied by Figure 5(a), but we have:

Lemma VI.2. The LR-pairs can be routed from the left and right edges using $n_{lr} + 1$ horizontal tracks, which is optimal.



(a) a single pair

(b) interleaving many pairs. x_i is in conflict with y_i (for $i = 1, 2, 3$).

Figure VI.5: Routing conflicting pairs in a T-shaped channel. (a) shows single pair, and (b) shows how to interleave many pairs.

Proof. The lemma is proven by the construction of Figure 5(b). Notice that nothing can be gained by interleaving the crossing points of different pairs. The construction can be described as follows: The signal corresponding to the L-term of the first pair is routed all the way to the rightmost vertical track, and jogs down; the signal corresponding to the R-term of the pair is assigned to the next higher track and jogs down immediately to the left of the former signal, thus forming the desired crossover. The second pair uses for its L-term the *same* track that was used for the former R-term, thus sharing a track, and the crossover is formed to the left of the former one, etc. Notice that (a) there is no significance whatsoever to which order the pairs are picked in, and (b) we could have likewise gone from left to right in the crossover ordering, putting signals for R-terms below the ones for L-terms.

The optimality is proven by induction. One pair needs two tracks for density reasons. Each additional pair requires at least one more track, again by a density argument. ∎

Notice that in the construction of Figure 5(b) one track was unused to the left of the leftmost crossing point, and another track was not used to the right of the rightmost crossing point. This free space is utilized in subsequent routes.

The next stage is the ordering of signals along the crossing edge. On the left we put those signals associated with L-terms that are not involved in conflicts with non-L-terms; likewise on the right. The LR-pairs are being put in the middle. M-terms not involved in conflicting pairs are routed straight down through the central portion. As for ordering on the left and right edges: Signals corresponding to LR-pairs are put as low as possible. This leaves one free horizontal track either on the right or the left (depending on the direction used in the construction of the crossovers, as in Lemma 2) which is used by one (any) signal of the corresponding side. All other signals corresponding to L- and R-terms are put above. This strategy yields the situation in Figure 6(a), which is abstracted schematically in Figure 6(b).

Thus, if we define

$$\chi_T = \begin{cases} 1, & \text{if } n_{lr} \neq 0 \text{ and } n_l = n_r = n_{lr} \\ 0, & \text{otherwise} \end{cases}$$

we have

Theorem VI.1. If $n_{ml} = n_{mm} = n_{mr} = 0$, then* the width of the top part of a T-shaped channel is $w_T = \max(n_r, n_l) + \chi_T + 1$.

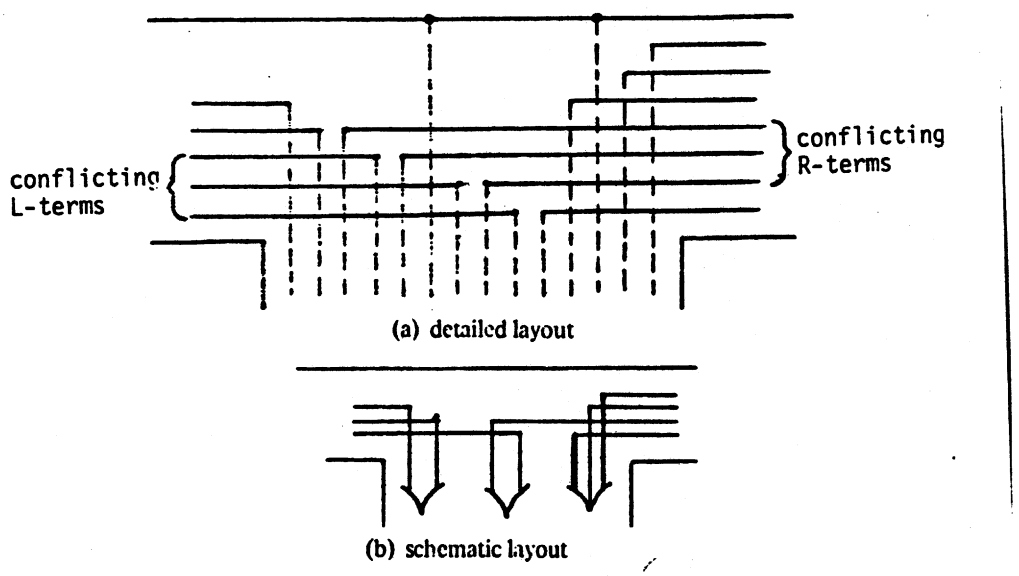


Figure VI.6: Routing in absence of conflicting M-terms. (a) is a detailed layout, and (b) is schematic diagram of it.

Proof. Conflicting LL-pairs and RR-pairs are ordered properly on the crossing edge. Horizontal tracks for signals of L-terms and R-terms are then assigned arbitrarily so as to form the situation in Figure 6(b). If $\chi_T = 0$ then either there are no LR-pairs to worry about, or the extra horizontal track needed to accommodate the LR-pairs in the central portion is being used by a signal either on the right or on the left. Only if $\chi_T = 1$, we are forced to use an extra horizontal track. ∎

Changing the wiring model to the knock-knee model, in which two signals may share a common turning point, does not affect this result.

Notice that we have opted to resolve conflicts only in the top part of the channel. Some such conflicts, however, could have been resolved in the bottom part without loss of optimality there. Since exploring this possibility complicates matters considerably, this has not been done.

* we add 1 at the end because w_T measures width, which is 1 more than the number of tracks.

The additional complexity does not justify the effort since the best it can help is by reducing w_T by 1 for Theorem 1 (only if $n_l = n_r$ and then by resolving all conflicts) or 2 for Theorem 2 (by resolving most conflicts). Thus, the notion of "near-optimal" in this section and in the subsequent one should be viewed realtive to this simplifying decision, but it is not far from being truly optimal.

The case in which M-terms are involved in conflicts is dealt with by extending the paradigm of Figure 6. First, ML- and MR-pairs whose M-terms are above the crossing points allocated for the corresponding L-term or R-term, respectively, can be accommodated in the appropriate ranges by simply forcing assignments of crossing points to the corresponding L- or R-term (e.g. D and F in Figure 7(a)). The number of ML-(MR-) pairs not handled in such a way is denoted by n'_{ml} (n'_{mr}), and consequently we define $n'_m = n_{mm} + n'_{ml} + n'_{mr}$.

Other conflicts of the above kind and MM-pairs are handled one at a time in the remaining tracks, making full use of track segments left free in the upper-middle part of the central portion. The block of LR-pairs is pushed all the way towards the more congested edge. A greedy approach in assigning tracks at this stage (on a pair-by-pair basis, putting the two crossing points as close to each other as possible) is good enough to attain a minimal w_T; again, pairs share tracks as LR-pairs did. The only trouble is with M-terms which are too close to the right and left edges and are involved in conflicts with L-terms and R-terms, respectively, or appear in MM-pairs. Surprisingly, this might cost us at most 1 extra horizontal track:

We say that an M-term is *m-adjacent* to the right (left) edge if all (possibly zero) grid-points between it and the edge (along the top side) have M-terms located at them. Now we define $\mu_T^R = 1$ if an M-term involved in a conflict with an L- or M-term is m-adjacent to the right edge *and* $n_r > n_l$; $\mu_T^R = 0$ otherwise. μ_T^L is defined likewise by reversing the roles of left and right in the definition. Also, $\mu_T = \max(\mu_T^R, \mu_T^L)$. Finally, $\phi_T = 1$ iff both left edge and right edge have m-adjacent M-terms involved in such conflicts *and* $n_r = n_l$.

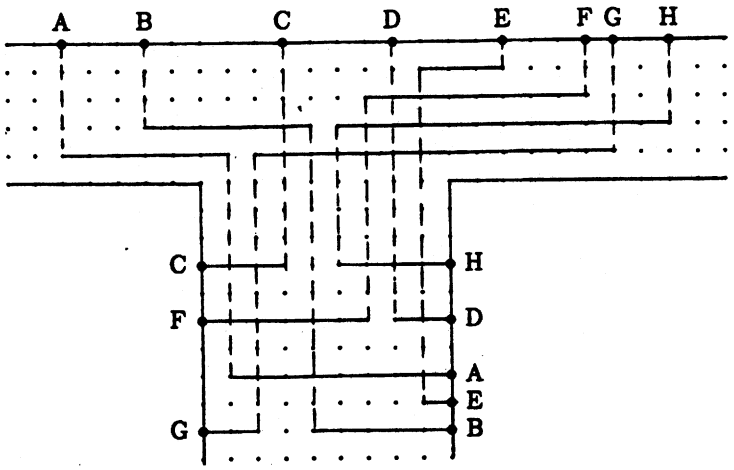
Now we are ready to state

Theorem VI.2. The width of the top part of a T-shaped channel is given by

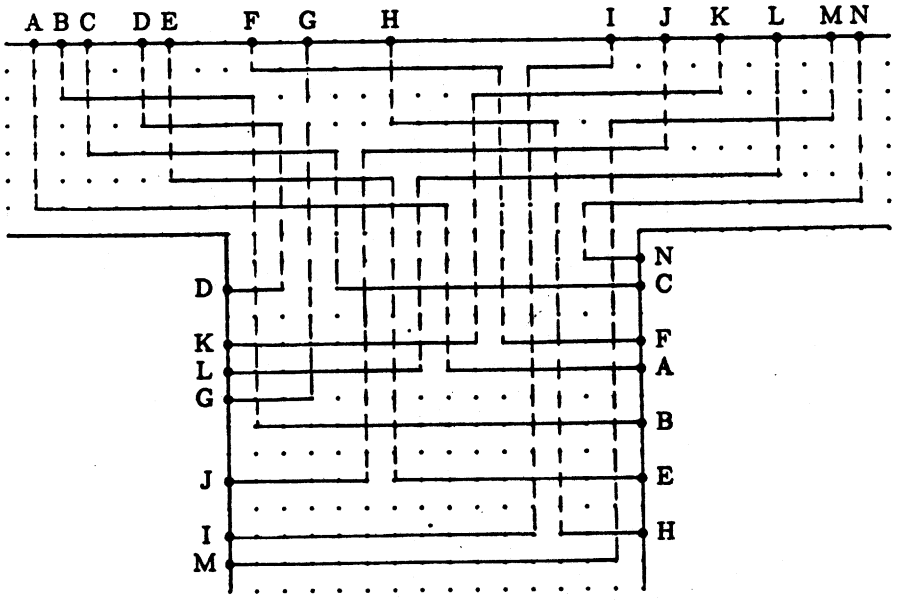
$$w_T = \max(n_r, n_l) + \max(\chi_T, \phi_T) + \mu_T + \max(0, n'_m - (n_{lr} - 1 - \min(n_r, n_l))) + 1.$$

Proof. The addend before last* is due to the fact that there might be too many MX-pairs (for X=M,L,R) to fit into previously allocated tracks, so the excess has to be allocated new tracks. Figure 7(b) shows an elaborate (but not exhaustive) case of routing in a T-shaped channel.

* the 1 at the end appears for the same reason as in Theorem 1



(a) All conflicting pairs involving M-terms are simple.



(b) A more complicated case, in which M-terms are involved in conflicts with L- or R-terms of the far portions.
Here $n = 14$ with $n_l = 5$, $n_r = 5$, and $n_m = 4$. Using the definitions given in the text, we obtain $\chi_T = 0$, $\phi_T = 1$, and $\mu_T = 0$ (since $n_l = n_r$). By Theorem 2, $w_T = 7$.

Figure VI.7: Complete routing in T-shaped channels.

If we use the knock-knee wiring model, μ_T and ϕ_T disappear from the sum in the statement of Theorem 2. Since their product is always 0 and their sum is at most 1, the effect of their omission is quite insignificant.

The dominant operation involved in the algorithms used to attain this optimum is sorting of the terminals, thus the complexity of both routing and finding the optimum is $O(n \log n)$, but the actual routing takes many more operations to complete. In case the terminals are presorted, which is common, the complexity is $O(n)$.

2. Pairwise Ordering and Routing X-shaped Channels

The major notion emerging from the previous section is that of sharing the constraints of two almost independent rectangular channel routing problems by an ordering of points on their common edge. This ordering is very sparse, but extremely powerful.

Definition VI.3. Given a set A, a *pairwise ordering* $\mathcal{W}$ of A's elements is a binary, antisymmetric relation over A such that if $(a_i, a_j) \in \mathcal{W}$ then $a_i \neq a_j$ and neither a_i nor a_j appear in any other member of $\mathcal{W}$.

The interpretation of $(A, \mathcal{W})$ as a directed graph induces an undirected graph with bounded degree 1 (see Figure 8(a)). The reason this algebraic structure represents channel routing constraints is that exactly two signals are involved in each conflict, no signal is involved in more than one conflict, and the conflicts are directional in nature.

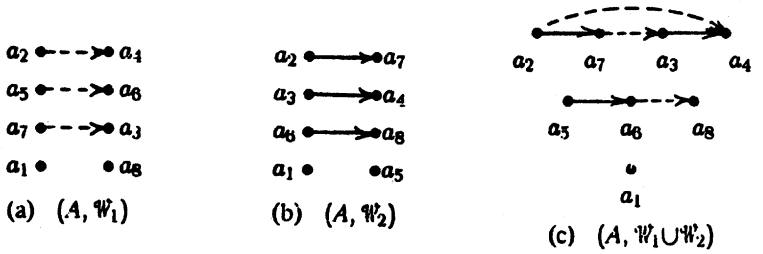


Figure VI.8: The graphic representation of pairwise orderings and their union. $A = \{a_1, \ldots, a_8\}$ $\mathcal{W}_1 = \{(a_1, a_4), (a_5, a_6), (a_7, a_3)\}$, and $\mathcal{W}_2 = \{(a_2, a_7), (a_3, a_4), (a_6, a_8)\}$.

An X-shaped channel (Figure 9) is most naturally partitioned into five portions: four arms and one central portion. If we ignore, for the time being, terminals on the channel's ends, the constraints propagating from the arms inwards to the edges separating the arms from the central portions are simply pairwise orderings. Again, if we restrict ourselves to two-point nets and ignore signal nets connecting terminals in the same arm, the central portion is a rectangular channel with pairwise orderings on its 4 sides. For sake of simplicity, we deal here only with nets having points on opposite edges (but not adjacent ones).

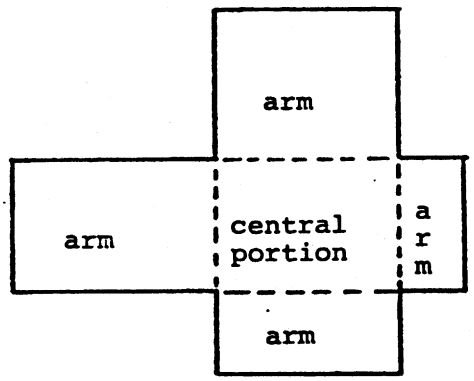


Figure VI.9: An X-shaped channel and its parts.

Each edge of the central portion has a pairwise ordering associated with it. In routing this portion, we must satisfy these constraints. Look at the structure obtained by taking the union of two pairwise orderings, $\mathcal{W}_1$ and $\mathcal{W}_2$, defined on the same set of elements, which is exemplified in Figure 8(b) and (c). The graph interpretation of the resulting structure induces an undirected graph with bounded degree 2, thus it consists of isolated vertices, open paths and even-length cycles. Open paths and cycles that are not directed cycles (*i.e.* there are at least two arcs going in opposite directions) can be arranged on a line such that all arrows go in one direction by topologically sorting the nodes (see Figure 10). Thus the union of two pairwise orderings corresponding to opposite edges of the central portion of an X-shaped channel can be arranged in such a way that signals in the central portion can go straight across unless there is a directed cycle.

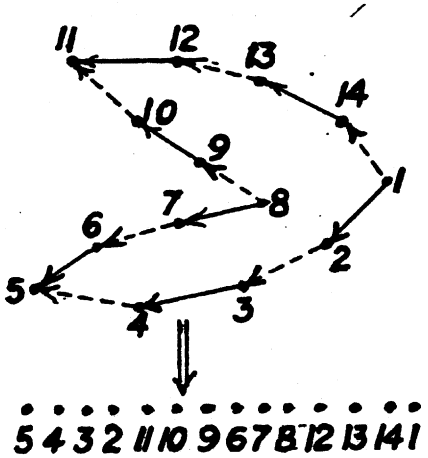


Figure VI.10: Topologically sorting a cycle that does not create a conflict.

Directed cycles are the only interesting case to look at. Obviously, a cycle involving k nets can be routed rectilinearly using $k+1$ tracks in one direction and 1 in the other (Figure 11(a)). This turns out to be optimal for one cycle, whether we are using the directoinal or the knock-knee wiring model. However, sharing tracks between cycles going in perpendicular directions turns out to be beneficial.

Before we proceed, let us introduce some notation. A cycle whose points are on the horizontal (vertical) edges is called a vertical (horizontal) cycle, because of the directions of the signals. The number of vertical cycles is denoted by v, and of horizontal ones — by h. We denote the total number of cycles by $c = h + v$. Obviously, we need at least h horizontal tracks and v vertical tracks to route across the central portion of an X. Let Δv and Δh denote the number of extra vertical and horizontal, respectively, tracks needed to do the routing.

Theorem VI.3. For the directional two-layer wiring model, $\Delta v + \Delta h = c + 1$. In the knock-knee model $\Delta v = \Delta h = 1$. Moreover, in the directional model, any pair of values for Δv and Δh satisfying $\Delta v + \Delta h = c + 1$ for $\Delta v, \Delta h \geq 1$, can be attained.

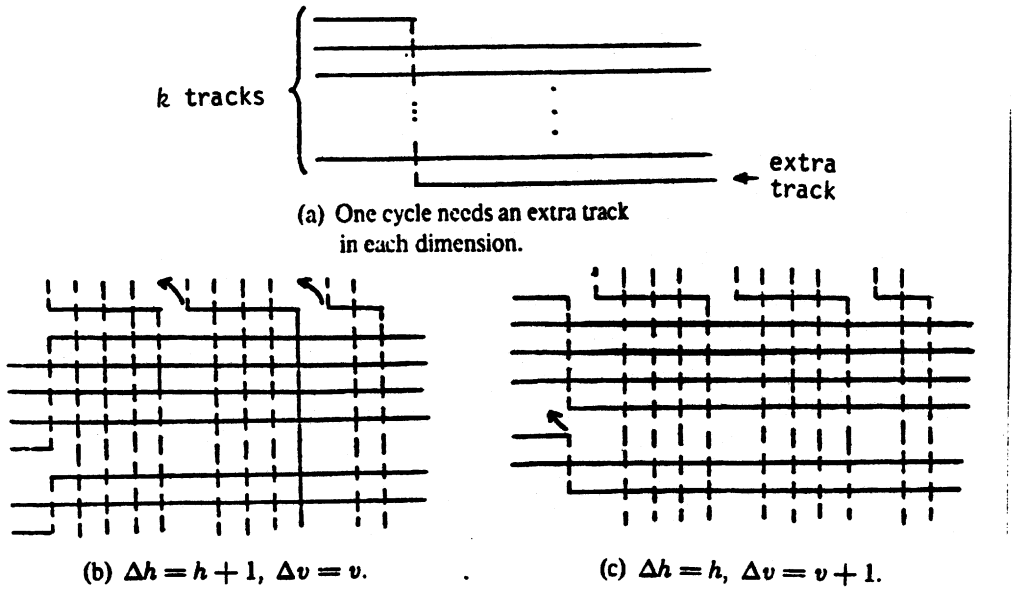


Figure VI.11: Laying out vertical and horizontal cycles in an X-shaped channel.

Here $h = 2$, $v = 3$.

Proof. The construction follows the paradigm of Figure 11(b) and (c). Both cases attain the claimed bound, and can be folded around ((b) horizontally and (c) vertically as indicated by the arrows) to attain all interim values. The result for the knock-knee model is achieved by merging corners. The optimality is proven by induction on c.

Using the result of Theorem 3, different optimality criteria can be employed to achieve desirable layouts.

3. Other Orderings

Pairwise orderings arise mostly due to the interaction between rectangular channel areas. Other orderings may arise due to such interaction, but more importantly, if we want to account for all of the routing done on a chip, we have to consider ordering requirements as they come off modules' sides. Such orderings have been encountered in previous chapters, but have not been explicitly handles as such. Here we shall name three such orderings, in preparation for the discussion in Section 4.

First we introduce the *total* ordering, which is essentially generalization of cables (that were discussed in Chapter IV). In this ordering a *total order* (in the algebraic sense) among the nets is specified, but no exact location is associated with each of them. This is typical for an an ordering that arises in global routing due to channel interaction, wether cable are being used or not.

Next we formalize orderings that come off modules sides. Such orderings have been discussed in Chapter V, and all we do here is recast their definitions in terms of orderings. A *fixed* ordering is a set of terminals along a straight line with an exact locaion associated with each. Thus the terminals induce a total order on nets, and they also have distances between their positions that have to be kept. This is the commom ordering arising from a side of a module.

An *arbitrary* ordering is a set of locations for terminals with no net designation for them. This ordering has been introduced in Section V.6, and occurs on sides of certain modules such as PLA's, I/O pads, and register arrays.

For completeness, we also introduce the *empty* ordering. This is simply a (not necessarily empty) set of nets with no relation between them. This is the counterpart of the arbitrary ordering when it comes to orderings that occur on channel boundaries. They arise as a result of propagating arbitrary orderings across channels, or when routing in an L-shaped channel, when any permutation of nets can be realized without extra area requirements.

4. Conclusions

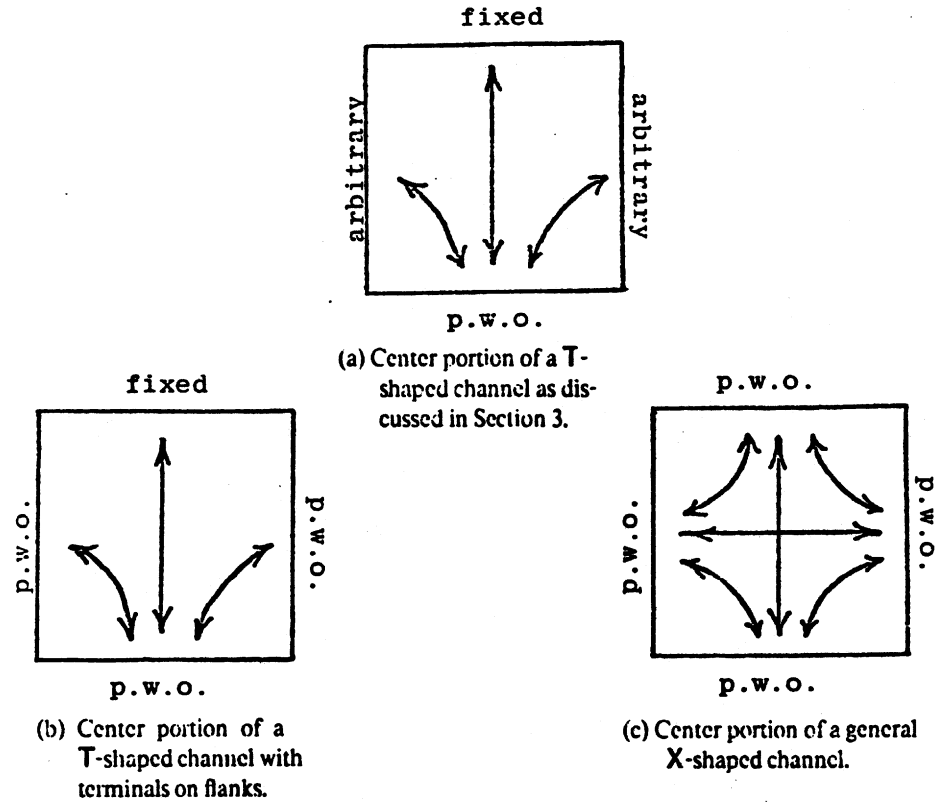
We have shown that optimal routing for some configurations in rectilinear-polygonal channels can be obtained efficiently. Technically some surprisingly compact, yet simple, routing patterns were discovered. Although most seem to be ad hoc, they share a common flavor which is induced

by the pairwise ordering introduced to represent constraint propagation. The results obtained are truly or nearly optimal, not just in order of complexity as in [Lei80].

Surprisingly, compact routing schemes can be achieved by decomposing the polygonal channels in a natural way, and solving the parts almost independently while maintaining constraints that are shared in a simple manner. This method gives rise to a general methodology according to which the routing area of a chip is divided into polygonal channels, which in turn are subdivided into rectangular parts. The original channels are used to form routing constraints in terms of orderings on the sides of these rectangles. The types of orderings on the sides of a rectangle and their interaction, in terms of common signal nets, induce a *typing* of rectangles. For example, the center portion of the T-shaped channel in Section 1 may be described as *arbitrary-fixed-arbitrary-pairwise* (going clockwise from the left edge) where nets are split between the first three sides and the last one (Figure 12(a)). Allowing terminals to reside also on the flanks yields a *pairwise-fixed-pairwise-pairwise* (with similar net splitting) description for the same portion (Figure 12(b)). An X-shaped channel in which two-point nets can be split in any way between two different sides yields a *pairwise-pairwise-pairwise-pairwise* description with the aforementioned net interaction (Figure 12(c)). Such types can be characterized in terms of the complexity of their optimal routing problem. Some, as we have seen, can be routed both optimally and efficiently, but other configurations may be intractable. Still, good heuristic solutions will be helpful.

For this method to be effective, we may need to allow channels to overlap, which relaxes the convention assumed so far. The common areas will reflect constraints arising from more than one polygonal channel which must be solved simultaneously. Trying to find independent solutions and piecing them together is probably a bad idea. Although this complicates matters slightly, the types of rectangles are essentially the same and the general methodology applies.

A further direction is to consider parametrized modules [Goo81] which can be integrated into the constraint propagation methodology to enhance the interrelation between placement and routing even further. Other interesting cases are skewed T's and X's (Figure 13(a) and (b), respectively) in which a side of an internal rectangle might be further subdivided. Solving the offset problem (generalizing the channel context as discussed in Section V.1) for such channels is another possible extension.

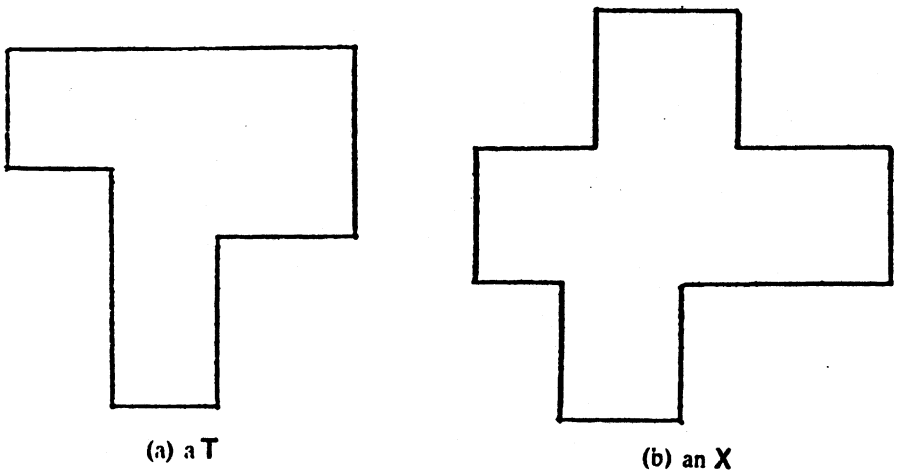


(a) Center portion of a T-shaped channel as discussed in Section 3.

(b) Center portion of a T-shaped channel with terminals on flanks.

(c) Center portion of a general X-shaped channel.

Figure VI.12: Possible routing configurations for rectangular parts of rectilinear channels. *p.w.o.* stands for pairwise ordering; a two-headed arrow indicates that routing occurs between the sides pointed at.



(a) a T

(b) an X

Figure VI.13: Skewed rectilinear channels.

Chapter VII

Layer Assignment for Interconnect

It has long been believed that optimizing layer assignment in itself is NP-complete even for restricted routing situations, and heuristics have been suggested to deal with the problem (traditionally, contact minimization is the objective and design rules are simplified). In this chapter we dispute this claim for the two-layer case. We show that achieving a variety of optimization criteria for the layer assignment problem (including contact minimization and metal maximization) for two-layer realizations subject to "real life" design rules *is tractable**.

We present a polynomial-time algorithm that accepts as input a purely geometric specification of the layout, an objective and design rules, and produces an optimal layout if it is possible or an indication that the design rules cannot be satisfied by the given geometry. As for three or more layer realizations, we conjecture that the problem is indeed NP-complete, based on our abstract representation of the problem.

In the following Section we give some more of the problem's background and an overview of the algorithm. For didactic reasons, we start off by presenting the formal definition of the *contact minimization* problem and its solution in Section 2. Section 3 explains how the same solution technique can be applied to solve the layer assignment problem in general. We conclude by listing open problems in Section 4.

1. Background

The *layer assignment problem for interconnect* is the problem of determining which layers should be used for wiring the signal nets. The goal is to optimize the performance of the circuit

* There is one rare special case that we cannot handle as yet, as we shall clarify in the technical part of this chapter.

and possibly minimize its manufacturing cost. The problem has appeared in different incarnations over the past dozen years.

The most popular version is the *via minimization* problem for printed circuit boards, and was first formulated by Hashimoto and Stevens [HaSt71]. They wished to minimize the number of via holes that must be drilled in a PC board in order to make connections between the two available conducting layers.

The development of the integrated circuit has added another dimension to the problem. The layers on an IC are not equal in performance. Deciding what portion of a wire should be implemented in which layer becomes important. The reason for reducing the number of *contacts* is to improve the yield of the fabrication process.

Another aspect of the problem is the design effort involved in specifying mask data. An efficient procedure for optimal layer assignment relieves the designer from an arduous task without compromising the quality of the resulting layout. In addition, she or he can use simpler equipment, such as a monochromatic or low reolution display.

In the literature published so far, only the contact minimization problem has been studied formally, and then only two layers were considered. Two recent papers on the subject obtained incomplete results. Kajitani [Ka80] shows how to use a polynomial-time algorithm for finding a max-cut in planar graphs (see [GaJo79], Section 4.1) to minimize contacts if they are allowed only at corners of paths (*i.e.* at places where a path changes its direction in a rectilinear design). In another paper, Ciesielski and Kinnen [CieKi81] give an integer programming solution to the case in which contacts are allowed wherever they fit, but the time complexity of their solution is exponential. In both these papers, the only way two paths may overlap is by simple crossovers.

In this chapter we present a polynomial time algorithm to solve the general layer assignment problem*. The solution proposed here is a three step reduction. First we two-color a graph representing conflicts between layers in the given layout. We then contract its connected components to single nodes assigning appropriate weights to the combined edges to represent the penalty associated with different coloring of components. Finally we solve the max-cut problem on the derived graph that always turns out to be planar.

2. Contact Minimization

We consider the following formalization of the contact minimization problem. "Given a collection of paths representing the interconnect wires without specifying what layers they use, find a layer assignment to all the paths segments that will minimize the total number of contacts

* Chen, Kajitani and Chan have independently obtained similar results for contact minimisation. It is not apparent that their algorithm, reported in the 1982 ISCAS Proceedings [CKC82], lends itself to dealing with metal maximisation or other optimisation criteria.

used." Pictorially, the problem setting is a colorless (black and white) picture of the interconnect pattern as in Figure 1, in which crossovers, overlaps, and sharing of corners are allowed. We are given a number of layers l, each in a different color, and ask "Can the picture be *colored* so as to minimize the number of points at which the color of a path changes along its way, and such that no two paths segments that use the same color are too close?" In this section, we constrain ourselves to two layers only ($l = 2$), and each wire is connected to exactly two points on boundaries of modules.

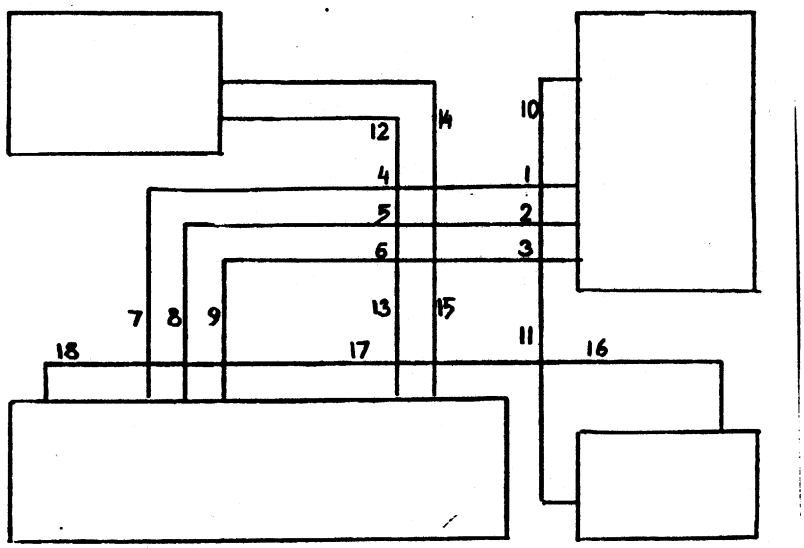


Figure VII.1: A "colorless" routing. The numbers along wires are for future reference.

First, we capture the notion of how different layer assignments to wires affect each other. It is clear that two wires that cross each other, overlap in some other way, or are even too close to be in the same layer, must be assigned different layers. On the other hand, we must consider the problem of whether there is enough room on a wire to place a contact where we may need it. These two problems go hand in hand in the sense that parts of wires that do not have room on them for layer changes because of overlaps must be assigned a single layer, and their assignment must be considered relative to the assignments of all conflicting wires. So we give the following complementary definitions.

Definition VII.1. A *free run* is a maximal piece of wire that does not overlap any other wire, and can accommodate at least one contact.

Definition VII.2. A *wire segment* is a piece of a wire connecting two free runs.

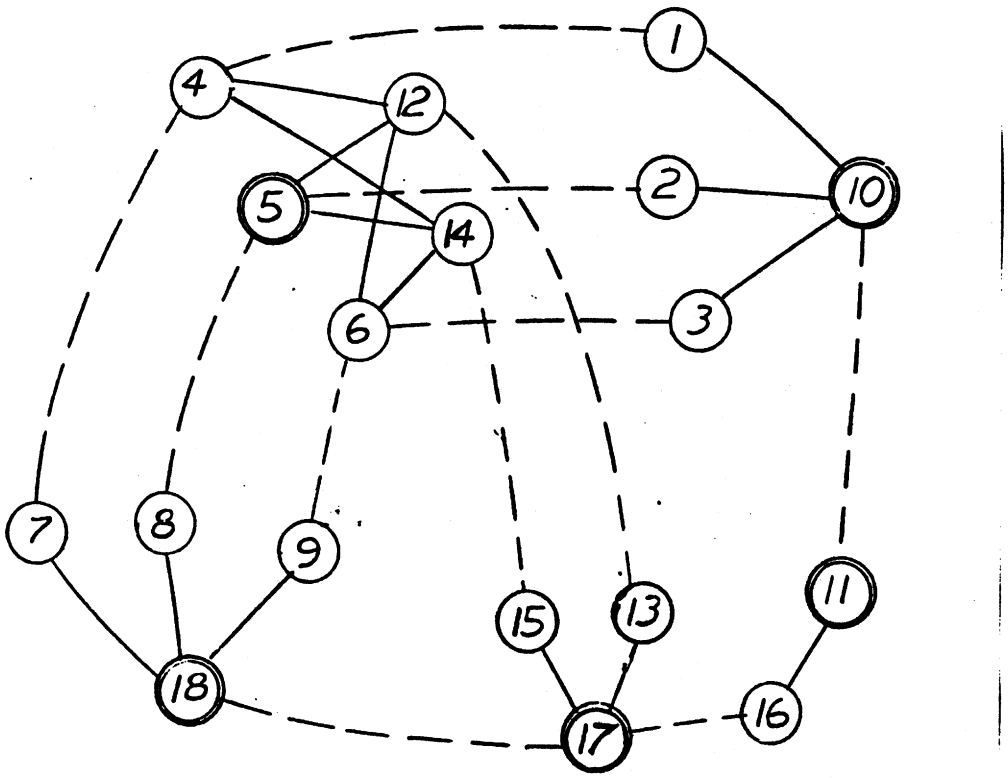


Figure VII.2: The layout graph for the routing shown in Figure 1. *Conflict* edges are solid (——), *continuation* edges are broken (— — —). Representative nodes are doubly circled for future reference.

The following graph structure encapsulates both the notion of conflicts between wire segments and the notion of how they are linked together, so that we can tell whether contacts are needed along them or not.

Definition VII.3. Let R be a given "colorless" routing. The graph $G(R) = (S, X \cup C)$ is the *layout graph* of R. Its nodes are the wire segments of R, denoted by S. Its edges are of two kinds — *conflict edges*, denoted by the set X, and *continuation edges*, denoted by the set C. These two sets are defined precisely as

$$X = \{(s_i, s_j) \mid s_i \text{ and } s_j \text{ cannot be on the same layer}\},$$

$$C = \{(s_i, s_j) \mid s_i \text{ and } s_j \text{ connect to the same free run}\}.$$

If $(s_i, s_j) \in C$ then s_i and s_j belong to the same wire. Figure 2 shows the layout graph corresponding to the layout of Figure 1. The numbers that appear next to wires in Figure 1 are the labels of the segments in S.

A segment must be assigned a *single* layer, because by Definitions 1 and 2 there is not enough room to put a contact along it. Thus, the problem of finding a feasible layer assignment for a

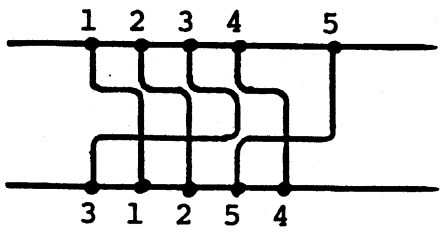


Figure VII.3: Routing across a channel in the knock-knee model may require three layers.

routing R is equivalent to the problem of coloring the *conflict subgraph* $G_X(R) = (S, X)$ of the corresponding layout graph $G(R)$. As long as the number of layers is only two, this coloring problem is easily solvable because in linear-time we can tell whether G_X is indeed two-colorable. Whenever it is, we obtain a way in which to color the nodes in S. If the conflict part of the graph turns out to be noncolorable with two colors, we can safely deduce that there is no way to assign two layers to the paths such that design rules are obeyed. Note that if the original routing was done using the directional model or by any other procedure that guaranteed that a two-layer realization exists, in which case the layer assignment algorithm can be just viewed as an improvement phase, there will be no problem in two-coloring G_X. The knock-knee model does not have the same property since it permits routings that are not two-colorable. Figure 3 shows that this may occur even when routing two-point nets across a channel. However, even if the directional model is being used we cannot skip the two-coloring stage because we need its results to solve the general problem.

The following observation concerns the identification of colors with nodes. For each connected component of the graph (G_X, in this case), all that the two-coloring algorithm produces is a partitioning of the nodes into two sets such that the nodes in each set are colored the same. The partitioning is unique for each component, but we have the freedom to decide which color to assign to which set. For each component, once a representative node is picked, the coloring of the rest of the nodes in the component is forced. We arbitrarily select representatives, one from each component, and number them uniquely. We denote the color of the ith representative by a binary variable, y_i.

The coloring of different components is independent insofar as the feasibility of the layout is maintained. Our goal is to find an assignment which will cause the minimum amount of color switching. To do this we need the wiring information as represented by the continuation edges, C. A simple observation is that whenever we decide to place a contact on a free run, one contact is enough because there is no sense in switching layers twice or more along a single free run, that is, alnog a continuation edge, if we are trying to minimize their number.

This is where the next step starts. We contract the connected components of G_X to single nodes, each labeled now by its representative. These connected components are the nodes of the weighted *residue graph*, $G/X = (V, E, \sigma, \delta)$, whose edges are defined as

$$E = \{(v_i, v_j) \mid \exists s \in v_i,\ t \in v_j \text{ s.t.} (s,t) \in C\}.$$

The edges can be thought of as contractions of the original continuation edges, C, in the layout graph G. The two remaining parts of the definition are both *weight functions* from the edges to the integers. With each edge we associate two numbers telling us how many color changes (contacts) will be needed if the representatives of the components incident upon the edge are assigned the *same* or *different* colors. Formally, for each $e = (v_i, v_j) \in E$, we define $\sigma(e)$ to be the number of color changes needed if $y_i = y_j$, and $\delta(e)$ to be the number of color changes needed if $y_i \neq y_j$. Figure 4 shows the residue graph for the layout of Figure 1, with representatives as shown in Figure 2.

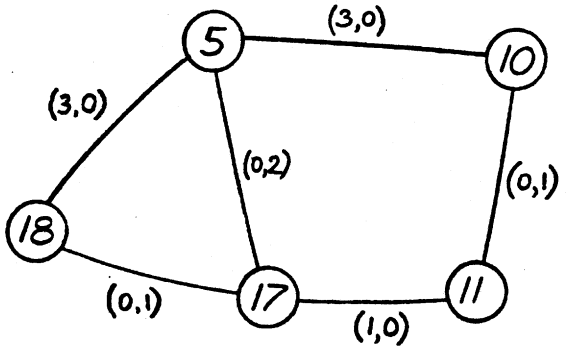


Figure VII.4: The residue graph corresponding to the layout of Figure 1, using the representatives as numbered in the nodes. A denotation (s, d) on an edge e means $\sigma(e) = s$, $\delta(e) = d$.

The weight functions σ and δ can be computed in time linear in the size of the original layout graph, G, by using the following technique. First we two-color all components of G_X arbitrarily with respect to relative colorings of components, which we know can be done in linear time. Now, for every edge $e = (v_i, v_j) \in E$, we simply check whether $y_i = y_j$. If yes, then we check all continuation edges going between segments of components i and j and count the number of edges that connect segments with the same color and the number that connect edges with different colors. The first number is set to be $\delta(e)$, and the second is set to $\sigma(e)$. If, on the other hand, $y_i \neq y_j$, then σ and δ change roles. All that matters in setting the weights is whether the components are colored compatibly or not, relative to their representatives, but we do not care about their particular colors. The choice of representatives is immaterial, as well, as long as the same representatives are used throughout the procedure.

Now we can state our goal of minimizing the number of contacts in the layout as

$$\min \quad z = \sum_{y_i = y_j} \sigma((v_i, v_j)) + \sum_{y_i \neq y_j} \delta((v_i, v_j)) \tag{1}$$

for all possible assignments of y_i's to nodes in G/X. To express the minimization goal in purely algebraic terms, we can associate numeric values with the predicates $=$ and $\neq$. We say that a predicate has the value 1 if it evaluates to true, and 0 otherwise. For instance, $(y_i = y_j)$ has the value 1 if y_i and y_j have the same value, and 0 otherwise. With this notation we can reexpress z using only one summation:

$$z = \sum_{(v_i, v_j) \in E} (\sigma((v_i, v_j)) \cdot (y_i = y_j) + \delta((v_i, v_j)) \cdot (y_i \neq y_j)).$$

The conditions $(y_i = y_j)$ and $(y_i \neq y_j)$ are mutually exclusive, and thus we can express one in terms of the other. By also relaxing the notation somewhat and understanding that $e = (v_i, v_j)$ whenever it appears, we obtain

$$z = \sum_{e \in E} (\sigma(e) \cdot (1 - (y_i \neq y_j)) + \delta(e) \cdot (y_i \neq y_j)).$$

But $\sum_{e \in E} \sigma(e)$ is a constant for the given graph, so we can ignore it and restate the goal given in (1) as

$$\min \quad z = \sum_{e \in E} (\delta(e) - \sigma(e)) \cdot (y_i \neq y_j) \tag{2}$$

for all possible assignments of y_i's.

Once we arrive at this stage, we can start the final step in our procedure. Remember that the y_i's are binary variables, but all that we want to know about any pair of them is whether they are different or not. Thus we certainly do not have to check all possible assignments of their values, we just have to look at all possible ways of putting these values into two disjoint sets; that is, all ways to partition the nodes of G/X. Such a problem, of finding a partitioning of the nodes of a graph, is a *cut* problem. In particular, we are interested in finding the *maximum* cut of the graph $G' = (V, E, \sigma - \delta)$ in order to solve the minimisation* problem as stated in (2). Figure 5 shows the optimal solution for the problem of Figure 1. It is obtained by the cut $V_1 = \{5, 17\}$, $V_2 = \{10, 11, 18\}$ in the graph of Figure 4. The cut value is 6, and $\sum_{e \in E} \sigma(e) = 7$, thus the number of contacts needed is 1.

In general, the max-cut problem for weighted graphs is NP-complete [GaJo79, Section 4.1], but now comes the final observation. The connected components of the conflict subgraph G_X of

* Notice that we now subtract δ from σ to match the maximisation goal with the original minimisation statement; this is to conform with standard terminology.

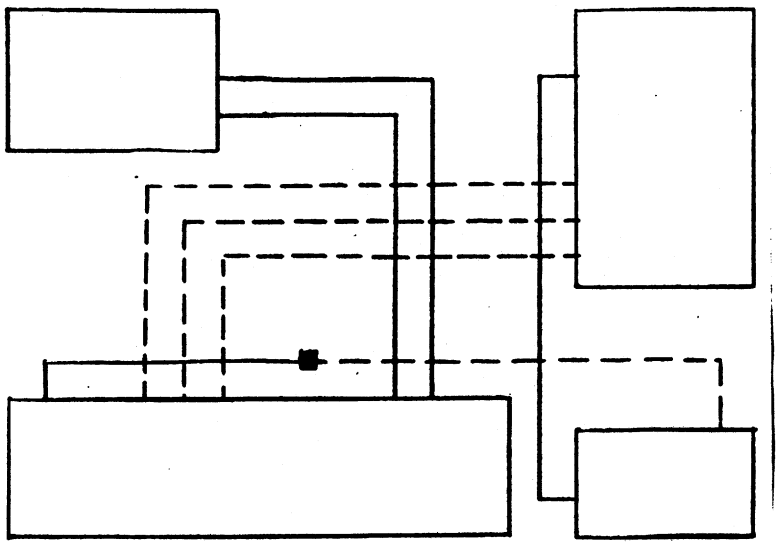


Figure VII.5: An optimal solution for the layout in Figure 1 using one contact. Layer 1 is drawn in solid lines (——), layer 2 is in broken lines (— — —), and contacts are solid squares.

the layout graph G, correspond to all neighbourhoods in which crossings can occur in the given geometry. Thus, the residue graph, G/X, which has one single node for each such neighbourhood, must be planar, as is G'. Fortunately, the max-cut problem for planar graphs with arbitrary weights (including negative numbers, as $\sigma(e) - \delta(e)$ may well be for some edges) is known to have a polynomial time solution ([OrDo72], [Ha75]).

The solution technique suggested in both these references is based on finding a maximum matching for a graph. Since these papers were written, some new results on matching have appeared, and it is entirely possible that they will be further improved in the future. So if we denote the running time of a matching algorithm on a graph with n nodes by $f(n)$, the running time of our algorithm for a given layout R is $O(f(|R|))$, where $|R|$ is the length of the input describing R. Since f is known to be at least linear, and all the steps we went through up to the max-cut step were linear in $|R|$, this statement is safe. The best value currently known for $f(n)$ is $O(n^{2.5})$ [MiVa80].

So, all in all, we have proven the following result.

Theorem VII.1. A given routing R can be assigned two layers so as to minimize the number of contacts used to connect between them in time $O(|R|^{2.5})$.

This result is independent of any particular design rules. All we need to know is how close wires of the same layer are allowed to get. Then we can abstract the appropriate graph from the

given colorless picture and apply the technique as described. The size of the description of the routing, $|R|$ may depend on the specific geometries used. For the common rectilinear style, $|R|$ is typically the number of corners in wires (counting endpoints as well).

3. Extensions.

The method proposed in the previous section is general enough to consider different performance of the two layers. That is, if we use, say, metal and polysilicon as our interconnect layers, we can maximize the amount of metal, or minimize a linear combination between the number of contacts and the penalty involved in using polysilicon. This problem has been dealt with only informally, so far, in the "PI" system [Ri82], but now the rigorous solution suggested by the new method will be incorporated in it.

The basic idea of metal minimization is quite simple. The weights σ and δ now reflect the relative merits of coloring one component in one way or the other. Even if these weights depend on the lengths of various wire segments and free runs, our technique still works. Notice that by trying to maximize the total length of metal wires we may incur more contacts. This problem can be remedied by finding a common scale to measure the "badness" of contacts and that of the nonpreferred layer, and then combining the two linearly. Again, this affects only the values of the weights, which must be multiplied by the scale coefficients, and the rest of the technique still holds.

In Section 2 we only showed how to handle two-point nets. This constraint can be relaxed almost completely. If a wire connects more than two terminals, it must have internal splits. Most commonly, we find only three-way splits, which can be modelled in $G(R)$. Replace each split by a constant number of edges and weights as shown in Figure 6. This replacement faithfully reflects the required number of contacts depending on the coloring of the segments that the edges connect. If the split is four-way (more is highly unlikely), we do not know how to handle it.

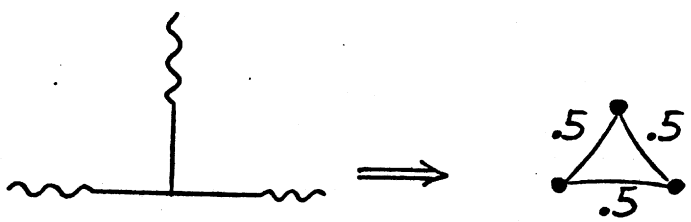


Figure VII.6: Transforming a 3-way split to the corresponding edges.

Another extension is very useful. Assume that the terminals are available in certain layers, but not in others. Connecting to a pin may then cost us a contact for which we also need to find a place to put it down. This problem can be solved using the above approach by inserting

extra edges with very large weights between terminals with different layers without destroying planarity. If the weights are polynomially bounded by the size of the problem, the solution is still obtainable in polynomial-time.

4. Open Problems.

Several problems remain open:

(i) What if $l > 2$? It seems that just the question of feasibility, *i.e.* whether a layout can be realized using l layers, is NP-complete, let alone minimizing the number of contacts (or the badness, in general). Even if, much to everybody's surprise, we could settle the feasibility problem efficiently, grave problems arise from allowing multiple contacts at one point, or even the coexistence of a contact between two layers and a wire in a third layer.

(ii) Our relaxation of the design rules is not entirely general in the sense that it does not reflect some of the more subtle aspects of the problem. There are two points to make here. First, the minimum center-to-center distance between wires varies from one layer to another. Before, we were just concerned with the question of whether two wire segments can be realized in the same layer or not. Now, we point out that the answer may depend on the actual assignment, which is not known at problem definition time. Second, in many fabrication technologies contacts are wider than wires. Putting a contact down may affect the allowable assignment in a neighbouring wire, and so again, a conflict depends on the solution, not just the naïve definition. These two manifestations of the problem of using full-blown design rules may break the symmetry exploited so strongly in the original solution, and may render the problem intractable.

(iii) Our solution is *global*. In the process of minimizing the total number of contacts we may burden one net with an excessive number of contacts. How do we minimize the *largest* "badness" figure of any wire in the layout?

(iv) As mentioned in the previous section, some geometries, such as four way splits in wires, cannot be handled properly. The only reasonable way to extend the analogy between the edges in the graph used in our technique and the allowability of contacts, seems to use hyperedges in a hypergraph. The question then would be, "What is a natural extension to the planarity criterion", and, more seriously, "Can we expect to solve the max-cut problem in polynomial time?"

Chapter VIII

Discussion

This thesis has investigated various aspects of layer assignment methods as they affect the design and complexity of layout algorithms for integrated circuits. We started off with a trivial layer assignment strategy, namely, routing in only *one* layer. From a relatively simple configuration of river routing across a channel that we showed to be solvable optimally and efficiently, we extracted a few essential properties that proved to be both interesting from a methodological point of view as well as rich in algorithmic content. In order to handle more complicated situations we used stronger strategies, both for layer assignment and for dealing with the interaction between routing areas, and the impact of these strategies on the quality of the resulting layouts were investigated. Finally, we turned around to ask how layers could be assigned retroactively so as to make a given piece of artwork realizable within certain design rules.

Let us summarise the characteristics of layer assignment methods and their relation to placement and routing strategies in light of the results presented in this thesis. At the same time we shall try to provide some useful pieces of advice both to circuit designers and to providers of design aids.

1. From River Routing to Arbitrary Interconnect

The term "river routing" embodies at least three important characteristics: all nets consist of exactly two terminals, the connections are made in order among two sequences, and the interconnection topology is realizable in the plane. Initially we imposed the pattern of going across a channel. This constraint was relaxed in subsequent discussions of river routing, but motivated a discussion of the nonplanar case in the channel context.

The first property of river routing that we generalized was the *sequence* compatibility. We noticed that terminals of certain two-point nets come off modules at both ends in similar sequences, *i.e.* they are ordered among themselves in the same way. This led us to the *cable* abstraction that was used to handle river-routing-like situations among distant modules on the chip. We showed how to test efficiently whether a cable was realizable in the plane.

The next characteristic that was extracted was the planar realizability for an entire design. We tried to fit many cables on one chip, first by allowing the modules of the design to be oriented arbitrarily and then by disallowing reflections. Finally, we investigated the geometric — as opposed to the topological — details of planar routing patterns within a given hole-free area that is not necessarily a rectangular channel.

River routing may be perceived as the ultimate push for planar realization of interconnect. Reaching the limits of this strategy led us to further pursue the implications of other layer assignment methods on the quality and feasibility of interconnect patterns. We investigated the impact of various wiring methods on routing across a rectangular channel, routing in junctions formed by the intersection of channels, and finally looked at layer assignment at the chip assembly level.

In studying the multilayer strategies, we started off by investigating some fundamental characteristics of routing *two-point* nets *across a channel*, thus forsaking the planarity assumption but staying within the channel context. Minimizing the width of the channel is traditionally the foremost concern. The attainable minimum turns out to be highly sensitive to the wiring model being used. We surveyed known results on the subject and introduced a new wiring model, the *via-free* model, and studied the minimum width problem within it. The major contribution of this part was the definition and investigation of new properties of routing strategies and their relation to the minimum width criterion. Other optimality criteria than width were suggested and discussed in the same framework.

Next the channel *structure* was generalized. Rather than dealing with rectangular areas, the routing in junctions where such areas intersect was studied. Based on certain two-layer routing models, we developed a methodology for generating and propagating routing constraints from terminals (on modules' sides) to the internals of channels. Routing configurations that arise from such constraints were classified, and optimal algorithms for some of these configurations were devised.

Finally, a broader aspect of the layer assignment problem for arbitrary interconnect patterns was discussed: given the geometry of some wiring and design rules for the artwork, the problem is to assign layers to features in such a way that some circuit performance function — such as signal delay — will be optimized. The complexity of such a problem depends qualitatively on

the *number* of layers that we are allowed to utilize, so the allocation of layers to routing should be a major consideration in the design process.

2. Design Considerations

What are the lessons of this thesis for an IC designer? Overall, there is the straightforward moral, "Find out what you get yourself into when making decisions about your layer assignment strategy." Such a strategy determines both the power of one's routing model, in terms of what routing situations can be handled, as well as the complexity of the combinatorial problems that arise when trying to generate the actual wiring. In this thesis we tried to establish some of these dependencies, and expose the implications of various routing paradigms. To be more concrete, let us spell out the dependencies between model selection and routing results as they should be reflected in the design process.

2.1. Cables and Cable Busses

One-layer realizations are limited in their capability of handling arbitrary wiring patterns. The interconnect graph must be planar, and if modules are not flippable, the situation is even more restrictive. On the other hand, the algorithms for testing whether the strategy is applicable for a given situation, as well as routing algorithms for producing truly optimal results themselves, are generally efficient with minor exceptions. Thus it is reasonable to spend time in trying to use a one-layer method before going on to more complicated strategies.

It is, however, naïve to believe that the interconnect of an arbitrary circuit could be realized in the plane. But this is exactly where our advice comes in. Because the actual routing is so much easier when doing it in one layer, *try to design in such a way that will make use of as many cables as possible*. This may not be so hard when designing certain kinds of circuits, such as *microprocessors*, in which a fair amount of the data is being grouped in sequences. In other cases, the order of signals coming off certain logical modules, such as PLAs, can be reordered at will (without changing the functionality of the circuit) so as to generate conformable sequences. We shall return to the subject of *ordering* later on in this chapter.

In general, it is hard to find a maximal subset of planar interconnections: this is equivalent to the *maximum planar subgraph problem*, and is known to be NP-complete [GaJo79, p. 197]. But the planarity of a single cable is easy to check both for orientability and connectivity. First, the proper orientation of a cable can be checked in constant time, so we have a fast way to detect planarity of such subgraphs. Second, it is easy to check whether the insertion of a new cable into an existing layout will violate its planarity or not. Thus a simple hill-climbing heuristic can be used with many start positions in trying to find a reasonably large set of cables all realizable in one layer. The reduction of complexity achieved by the cable abstraction can, in fact, be exploited

by any other heuristic for the maximal planar subgraph or even by an exhaustive procedure if the number of cables is small enough.

To the provider of support software for computer aided design (CAD), our advice would be to make cables "first-class citizens" in the design environment. The user should have at her/his disposal easy ways to specify and manipulate cables. Such aids should include operations for handling connectivity as well as tests for orientability and routability. In this way, the designer can test the feasibility of her/his design while creating it, exploiting planarity as one goes along, without the fear that at the end of the process she/he will find out that the planar realization failed without being able to trace it to a specific culprit.

Similarly, one would like to have support for parts of the design as well. We may partition the nets into sets that will be each realized in one layer. The planarity tests will apply to one set at a time, and the user should be able to specify the identity of such sets. There should be an easy way of moving around members of these sets without changing their connectivity data, but in a way that will make the necessary geometric adjustments.

Another consideration that is relevant to the design of microprocessors is the possibility that some of the signals "traveling" on a cable need to be fanned-out to more than one terminal. These are special kinds of multipoint nets that occur rather frequently in structured design such as microprocessors. We propose a *generalization* of cables that encapsulates this routing situation, namely *cable busses*. A cable bus is a construct in which two identifiable sequences must be connected in one layer, and some of the nets have more connections other than within the sequences. These extra connections could be realized by attachments to the main cable that are river routed in themselves on another layer.

Cable busses pose quite a few problems, such as how to bias the routing of the main cable so as to make the routing of the attachments easier, and how to conduct detailed river routing to targets that are "floating" along a wire, rather than being fixed terminals. These problems remain open, but the methodological framework exists for some support at the design aids level.

We summarize by giving a list of "rules" for good design in one layer:

(1) Use cables, since they reduce complexity.

(2) Provide design aids that know about cables and can manipulate them well.

(3) Assign the same layer to cables that run in "parallel".

(4) Provide subsetting facilities that maintain association with layers.

(5) Extract cables from multipoint nets if they can be used in a natural way.

2.2. Channel Routing

Obviously, we have to start by mentioning that river routing across a channel is tractable and lends itself to various placement considerations. So this is one routing model that can be analyzed accurately. If the channel can be river routed in a given area, there is no reason to go to another model. If, on the other hand, either the topology or the geometry are not adequate, we have to resort to other routing strategies.

The quest is usually minimizing the width of the channel. Different wiring models affect the quality of the attainable results. Other considerations sometimes play an important role as well. The lessons concerning channel routing can be summarized briefly as follows:

(1) The Manhattan model may cause the width of a channel to far exceed its density, and thus the knock-knee model should be used (in a guarded way, such as in [RBM81]) if one's Manhattan based heuristic is doing poorly.

(2) Heuristics should allow unlimited jogging in all two-layer models. The effect of bounding the number of jogs on the channel width can be disastrous — at least in extreme case, and no "average case" behaviour is known on the subject.

(3) Vertical nonmonotonicity is not useful whereas horizontal nonmonotonicity can indeed be helpful in reducing the channel's width.

(4) Vertical conflict cycles are easy to resolve, and there is no reason to exclude them from allowable routing patterns, as some routing packages do.

(5) If the genus of the interconnect pattern is low (less than the number of available layers), via-free routing is a good strategy, although it may require large width.

2.3. Channel Intersection and Ordering

Ordering between wires as they are routed globally (as opposed to channel by channel) may affect the layout quality quite significantly. Some of the ordering constraints arise from modules' sides, and others are a result of the interaction between sides as induced by the wiring model. The designer as well as the design aids provider should be aware of the impact caused ordering, especially at channel junctions. Solving large systems of routing constraints as they are mandated by orderings seems to be a computationally intractable task, as we have learned from other constraint propagation problems. Thus one should be able to manipulate orderings explicitly — either at the module edges, using parameterized modules, or at channel boundaries.

It is unclear how such an abstraction can be made successfully in a design aids environment, but the new breed of "expert" routers that are coming into being [Bat82] could have the potential of incorporating such an Alish notion into them.

2.4. Global Layer Assignment

Some of the effort invested in the specification of artwork — whether manually or automatically — is spent on the assignment of layers to various pieces of the interconnect. Having to decide which layer a wire will be implemented in as part of the path specification seems to bind a somewhat independent decision with the path selection mechanism. Layer assignment should be regarded as an end rather than part of the means during the routing stage. Although the layer assignment decisions may be relevant, they are distracting and tend to be made locally rather than globally. A beautiful example for doing away with layer assignment considerations is the Manhattan model. By obeying the routing rules that allow only simple crossovers, layers can be assigned retroactively without fear of unfeasibility. In more involved models such safety is not guaranteed, but since layer assignment can now be done efficiently (Chapter VII), the risk may be worthwhile.

Our advice to the designer would be to concentrate on the geometry, not the layer assignment. If there seems to be enough space, using the Manhattan model is preferable. If not, knock-knees and overlaps can used, but with caution. Do not use too many of them in congested areas, and particularly avoid impossible patterns (as in Figure VII.3).

For the design aids provider, our first advice is to facilitate "colorless" (black and white) routing. Let the user specify paths without selecting layers for them. In addition, supply a layer assignment service; the algorithm of Chapter VII will produce optimal results, but its running time of $O(n^{2.5})$ may be too slow for some situations. Fast heuristics, like those suggested in [CieKi81], may prove more useful from a practical standpoint. More subtle is the following request. The user should have an easy way by which she or he can remedy an infeasible situation. If the artwork is inconsistent with the feasibility of layer assignment, some stretching of the interconnect area may be required where contacts could be accommodated. Such stretching facilities should know about the interconnect being realized in that they will maintain connectivity of wires and will not interfere with the layout of neighbouring logic or completed routing.

As to the subject of using more than two layers, there is not much that we can firmly substantiate by formal results. Currently we do not even have proper modeling for all subtleties of the design rules.

Naturally, one would like to observe a reduction in routing area when more layers become available. But the complexity of the layout problems, such as layer assignment, is presumably harder. This should provide an incentive for working on heuristics to solve layout problems. In the meantime, we can suggest a few "high level" strategies which subdivide the layers into groups according to functionality. We dedicate one of the layers, either the top one or the bottom one, but *not* any of the middle ones, to one-layer routing of cables or other planar structures, and

then use the other layers in a general multilayer routing procedure. This strategy cuts down on the interaction between layers, and enables us to control the complexity of the process. The discussion of planar subembeddings and cable busses in Subsection 2.1 is an example of such a strategy.

Such a "functional partitioning" approach could also be used to some extent when performing retroactive layer assignment, as in Chapter VII. Now we try to extract planar subgraphs, and then try to two-color what is left. This may be too weak a strategy for congested interconnect patterns and may miss potential solutions all too often. But when the designer is made aware of the layer assignment strategy, such an approach may work.

3. Routing versus Routability: Impact on Placement

As has been pointed out throughout this dissertation, there is often a quantitative difference in terms of time complexity between the *routability* question and an actual routing algorithm. Sometimes it is faster to check whether a set of nets is routable within a given area than it is to produce the actual paths taken by the wires. For the problem of river routing across a channel, for example, we saw that testing routability takes time $O(n)$ for n nets, whereas producing the layout may take time $\Omega(n^2)$.

This differentiation has a major impact on placement evaluation. If a placement can be tested for routability relatively fast, it may pay off to invest more in this part of a placement procedure than is customary. In some extreme cases, as river routing across a channel in the presence of stretch lines, we saw that the routability conditions were integrated into the placement evaluation problem to yield *optimal* results. Although this is no doubt a rare case, the lesson to be learned is that routability in itself is of utmost importance. The discussion of routing in junctions demonstrated some of these characteristics as well.

Designers may still construct modules and assemble floor plans by hand. The layout, however, should be specified in such a way that *stretching* of modules and *moving* them around will be an easy task. The DPL/Daedalus [BMSSW81, Sh82] system encourages such a modular design style by referring to layout objects *relative* to each other.

Providers of design aids should be aware of the potential speedup gained by testing routability rather than instantiating a router. If you are writing a placement algorithm, be more inclined to check routability in certain cases than in others. More importantly, provide explicit aids for routability testing for the designer. Checking the feasibility of interconnect situations as they are being set up could be extremely useful.

4. Conclusions

The main thrust of this thesis was to find out how various *design methodologies*, especially layer assignment strategies, affect the *complexity* of layout algorithms, mainly for the routing task. In some cases, like the data-path construct, a thin line has been drawn between efficient, optimal solutions and intractability. In other case, the gap between P and NP has been made somewhat narrower by showing that certain common situations could be handled efficiently although their generalizations are intractable (or unclassified).

Appendix A

Glossary of Problems

This appendix contains a list of the problems discussed in the thesis. We adopt the problem statement form from [GaJo79], summarize the known results about each problem under the heading *"Status"*. The order of the problems in this list is more or less the same as the order in which they have been presented in the thesis, thereby trying to follow the development of ideas and refinement of situations as they evolved.

All problems are formulated in their decision version. For those problems that are solvable in polynomial time, there may be a qualitative difference between the time it takes to decide the problem and the time it takes to produce an actual solution (in the affirmative case). Such distinctions will be pointed out when applicable.

[RR] **RIVER ROUTING ACROSS A CHANNEL**

INSTANCE: Two increasing sequences of terminals, $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, on two parallel straight lines located at distance (separation) t from each other.

QUESTION: Can the terminals be connected *in order*, i.e. a_i to b_i for $i = 1, \ldots, n$, within the area confined between the two lines subject to rectilinear wiring rules?

Status: Doable in linear time (Section III.1). If we are interested in actually producing the layout of the interconnect, we can run the greedy algorithm (Section III.1) whose time complexity is $O(n^2)$. This can be improved by running the modified greedy algorithm (Section V.4) that runs in time $O(j)$, where j is the minimum number of *jogs* required to realize the layout. There are cases, however, where $j = \Omega(n^2)$ (more on this subject can be found under MJ). The routability result can be generalized to various other wiring rules at the expense of the complexity, which, however, remains polynomial (in n and t).

[PRR] **PLACEMENT FOR RIVER ROUTING (ACROSS ONE CHANNEL)**

INSTANCE: Two sequences of modules to be placed across a channel in such a way that their sides facing the channel are lined up straight (for each of the two sets), and that the interconnect pattern they set up is a river route (as in RR). Each module has a length, and the positions of terminals along the side of each module is fixed. We are also given two numbers, t (the separation, as in RR), and s (the spread).

QUESTION: Let $p_1, \ldots, p_{m_1}$ be the positions of the left corners of the modules on the top line (from left to right), and $q_1, \ldots, q_{m_2}$ are the positions for the bottom line. Is there an

assignment to the p_i's and q_j's such that the channel is routable within dimensions t and s subject to rectilinear wiring rules and so that the modules do not overlap? *i.e.* can we satisfy $\max(p_{m_1} + l^T_{m_1}, q_{m_2} + l^B_{m_2}) - \min(p_1, q_1) \leq s$ subject to the design rules, the bound t on the separation, and the colinearity restriction (where l^T_i is the length of the ith top module, l^B_j — that of the jth bottom module)?

Status: Can be solved in linear time, *i.e.* time $O(n+m)$ where $m = m_1 + m_2$ (Sections III.2, III.3). The time remains linear even when the actual placement is required. The complexity is polynomial for other wiring models as well, and is related to the corresponding result for the previous problem (RR). A special case of this problem is the *offset* problem suggested in [DKSSU81], which is obtained by setting $m_1 = m_2 = 1$.

[PDP] PLACEMENT FOR A DATA PATH

INSTANCE: A set of modules arranged in $k+1$ rows, with all modules in the same row having the same height. Each module (except those on the first and last rows) faces two channels — one at its top, the other at its bottom, with terminals on those sides. The top row has terminals only on the bottom of modules, the last — only on their top. All k channels are river routable, as in RR (so the *order* of modules in each row is fixed). The dimensions of the total routing area are also given: the sum of the widths of all channels, t, and a spread, s.

QUESTION: Is there a placement for the modules (in straight rows) such that all channels are routable, the sum of their widths does not exceed the prespecified total width t, and the total extent of the layout in the horizontal dimension (*i.e.* the distance from the leftmost edge of a module to the rightmost edge) does not exceed s?

Status: The problem is NP-complete in the strong sense (Section III.6). It remains so even if each module has the same number of terminals on both sides, *i.e.* nets just "feed through" modules (this is a common practical case for data path design). The fastest known method to solve the problem runs in time $O(n^k)$, where n is the total number of nets to be interconnected. If the widths of individual channels $t_1, \ldots, t_k$ are given (rather than a total width t), the problem becomes solvable in time $O(n^2)$ for rectilinear wiring rules. The time complexity in the latter case remains polynomial even if the wiring rules are changed, but the degree will change accordingly (depending on the corresponding results for RR).

The last two problems (PRR and PDP) could be formulated in terms of *stretch lines* rather than as placement problems. The formulation is easier in terms of modules, but in some design contexts (such as datapath design) it is more natural to view the problem as a *stretching problem*. In such contexts the placement issue may be disguised as a routability question, but stretchable modules do indeed set up (albeit restricted) placement problems.

[CR] CABLE ROUTABILITY

INSTANCE: A placement for a set of rectangular modules within a bounding box, two (not necessarily disjoint) sides of modules each containing a sequence of n terminals numbered in order from 1 to n.

QUESTION: Is there a one–layer rough routing for this configuration?

Status: The reachability aspect is solvable in time proportional to the number of modules. The orientability aspect is solvable in constant time (Section IV.2.1).

[PO] PLACEMENT ORIENTATION

INSTANCE: A set of (non-oriented, flippable) modules, terminals on modules' boundaries. Each net consists of two points.

QUESTION: Is there an embedding of the modules in the plane such that a one-layer rough routing is (topologically) feasible?

Status: Solvable in linear time, using your favorite planarity testing algorithm (Section IV.2.2, using [Ev79, Chapter 8]).

[PR] PLANAR ROUTABILITY

INSTANCE: A placement of a set of rectangular modules within a bounding box, terminals (of two-point nets) on modules' boundaries.

QUESTION: Is there a one–layer rough routing for this configuration?

Status: Solvable in linear time, by a modification to the Hopcroft and Tarjan [HoTa74] planarity testing algorithm (Section IV.2.3).

[DPR] DETAILED PLANAR ROUTABILITY

INSTANCE: A placement of a set of rectangular modules within a bounding box, terminals (of two-point nets) on modules' boundaries.

QUESTION: Is there a one–layer detailed routing for this configuration?

Status: NP-complete, by an adaptation of a result due to Kramer and van Leeuwen [KrvanL82] (Section IV.4.2).

[DPRH] DETAILED PLANAR ROUTABILITY GIVEN A HOMOTOPY

INSTANCE: A placement of a set of rectangular modules within a bounding box, terminals (of two-point nets) on modules' boundaries, homotopy (rough routing) for each net.

QUESTION: Is there a one–layer detailed routing for this configuration?

Status: Open.

[DRSRP]DETAILED ROUTING IN A SIMPLE RECTILINEAR POLYGON

INSTANCE: A simple rectilinear polygon with terminals (of two-point nets) on its boundary.

QUESTION: Is there a one–layer detailed routing for this configuration?

Status: Solvable in time $O(n+p)$, where n is the number of nets and p is the number of corners in the polygon (Section IV.3). If a layout is required it takes time $O(n^2 + p \cdot n)$.

[MCW2] MINIMUM CHANNEL WIDTH FOR TWO-POINT NETS (ARBITRARY INTERCONNECT)

INSTANCE: Two rows of terminals on parallel lines, each containing one terminal of each net, but the order is not necessarily the same. The distance between the lines is w, their extent is e.

QUESTION: Can the channel be routed in its given dimensions, w and e? *i.e.* can the nets be interconnected subject to the wiring rules in the given area?

Status: The only solid results on this subject are for two extreme cases. First, if the via-free model is used the problem is solvable in time $O(n^2)$, where n is the number of nets (Sections III.5, V.2.2). However, unless the channel is river routable, the minimum number of layers required to realize the interconnect must be used to guarantee minimum width with respect to the given number of layers. If we are free to use more than the minimum the problem is open. see also MLC. Also, if the configuration is a river route, the complexity can be improved to linear time (Section III.5, [Bar81]). On the other hand, in the general case, if the Manhattan (directional) model is used, and in addition, each wire is constrained to have at most one horizontal segment, then the problem becomes NP-complete [LaP80a]. (In the absence of vertical conflicts, using

the same routing strategy, the problem becomes solvable in polynomial time.) All other cases are wide open: when doglegs are allowed, or if the wiring model is more general, the problem remains unclassified. Szymanski has shown [Sz81] that using four-point nets makes the problem NP-complete even when doglegs are allowed.

[MLC] MINIMUM NUMBER OF LAYERS FOR A CHANNEL

INSTANCE: As in MCW2, and a number of layers, l.

QUESTION: Can the channel be routed using l layers in the via-free model? Can it be done in width w within the extent e? *i.e.* if each wire is routed in exactly one layer, are l layers enough to realize the permutation? If so, are w tracks enough?

Status: The first question can be solved in time $O(n \log l)$, where n is the number of nets (Section V.2.2). The problem becomes NP-complete if single-sided connections are allowed. The width question can be solved only if the number of layers used is the minimum required, and then the complexity rises to $O(n^2)$ for the rectilinear wiring model.

[VM] VERTICAL MONOTONICITY

INSTANCE: Two rows of terminals on parallel lines, each containing one terminal of each net, but the order is not necessarily the same.

QUESTION: Can the minimum channel width be attained by using only vertically monotone wires?

Status: Almost. One extra track may be necessary in order to avoid vertically nonmonotone wires altogether (Section V.3). Notice that we do not have to find the minimum width of the channel (see MCW2); we are only asking whether the minimum is attainable using the specified strategy.

[MJ] MINIMUM JOGGING

INSTANCE: Two rows of terminals on parallel lines, each containing n terminals — one of each net (the order is not necessarily the same), and a number k.

QUESTION: Can the minimum channel width be attained by using at most k jogs per wire?

Status: Open. In Section V.4 example are given for which the number of jogs for at least one net is $\Omega(\sqrt{n})$ in both the Manhattan and knock-knee models. This suggests that the question is difficult. We could also ask a similar question about the *total* number of jogs. Both questions were fully answered for the river-routing case: an $O(\max(n, j))$ algorithm was given in Section V.4 that would river-route a channel using j jogs if possible; the same algorithm can be used to test whether any net exceeds k jogs in time $O(\max(n, nk))$.

[CRC] CONFLICT RESOLUTION WITHIN A CHANNEL

INSTANCE: Two rows of terminals on parallel lines, each containing one terminal of each net, but the order is not necessarily the same.

QUESTION: Can the routing be completed within the span of the nets, *i.e.* with no wire exceeding beyond the rightmost or leftmost terminal?

Status: Solvable in linear time (Section V.5).

[RTC] ROUTING IN A T-SHAPED CHANNEL

INSTANCE: A T-shaped channel with terminals on its sides.

QUESTION: Can the nets be routed within the given area?

Status: This is obviously a generalization of MCW2, so the results from there apply here. One special case that was studies is when all nets have two points with one terminal on the top side and one on either side of the bottom (see Section VI.1 for terminology): then the problem is solvable in $O(n \log n)$ time (where n is the number of nets) in both the directional and knock-knee models provided only one jog per net is allowed in the bottom part of the channel.

[RXC] ROUTING IN AN X-SHAPED CHANNEL

INSTANCE: An X-shaped channel with terminals on its sides.

QUESTION: Can the nets be routed within the given area?

Status: This is a generalization of RTC. Again, there is one special case, namely when all nets have two points, split between opposite flanks (see Section VI.2 for terminology): then the problem is solvable in time $O(n)$ in both the Manhattan and knock-knee models.

[RAT] ROUTABILITY OF AN ARBITRARY TILE (WITH ORDERS)

INSTANCE: A rectangle with one terminal* set associated with each side. The terminals in each set have a relation defined among them that partially specifies the order in which they are allowed to appear along the side. Some terminals may be associated with *fixed* locations along their sides.

QUESTION: Is there an assignment of terminals to locations that will conform with design rules and make the channel routable?

Status: Open. One simple case that has been solved is when two-point nets are split across a channel, and terminals are allowed to be at fixed locations. Then it takes time $O(n)$ to route the channel or to test it for routability (Section V.6).

[LAI] LAYER ASSIGNMENT FOR INTERCONNECT

INSTANCE: A layout including the *geometry* of the interconnecting wires, and a set of design rules (including the number of layers, l). In addition, given is a set of "badness" coefficients $b_1, \ldots, b_l$ for the layers, one coefficient c for contacts (layer changes), and a total figure, B.

QUESTION: Is there an assignment of layers to the wires, such that if the total length of wires in layer i is x_i, and the total number of layer changes is v, then $c \cdot v + \sum_{i=1}^{l} b_i \cdot x_i \leq B$?

Status: Solvable in polynomial time for $l = 2$ as long as there are no four-way junctions (Section VII.2). The problem for $l \geq 3$ is conjectured to be NP-complete. The general two-layer case (allowing four-way junctions) is open.

* We use the term *terminal* in a slightly different way than usual, namely now it is not necessarily associated with a fixed location, it is just a member of a net.

References

[Ak72] Akers, S. B.: "Routing"; in *Design Automation of Digital Systems: Theory and Techniques,* Prentice-Hall, 1972 (M. A. Breuer, *ed.*), Chapter 6.

[AGR70] Akers, S. B., J. M. Geyer, and D. L. Roberts: "IC Masks Layout with a Single Conductor Layer"; *Proceedings of the Seventh Design Automation Workshop,* June 1970, pp. 7–16.

[Bar81] Baratz, A. E.: *Algorithms for Integrated Circuit Signal Routing* (Ph.D. dissertation); Dept. of Electrical Engineering and Computer Science, M.I.T., August 1981.

[Bat82] Batali, J.: *private communication,* May 1982.

[BMSSW81] Batali, J., N. Mayle, H. Shrobe, G. Sussman, and D. Weise: "The DPL/ Daedalus design environment"; *Proceedings of the International Conference on VLSI,* Univ. of Edinburgh, August 1981, pp. 183–192.

[BrRi81] Brown, D. J., and R. L. Rivest: "New Lower Bounds for Channel Width"; *Proceedings of the CMU Conference on VLSI Systems and Computations,* October 1981, pp. 178–185.

[Bu82] Burstein, M.: "A Non 'Placement/Routing' Approach to Automation of VLSI Layout Design"; *Proceedings of the 1982 International Symposium on Circuits and Systems,* May 1982, pp. 756–759.

[CKC82] Chen, R. W., Y. Kajitani, and S. P. Chan: "Topological Considerations of the Via Minimization Problem for Two-Layer PC Boards"; *Proceedings of the 1982 International Symposium on Circuits and Systems,* May 1982, pp. 968–971.

[CieKi81] Ciesielski, M. J., and E. Kinnen: "An Optimum Layer Assignment for Routing in ICs and PCBs"; *Proceedings of the Eighteenth Design Automation Conference,* June 1981, pp. 733–737.

[Deu76] Deutsch, D. N.: "A Dogleg Channel Router"; *Proceedings of the Thirteenth Design Automation Conference,* June 1976, pp. 425–433.

[DKSSU81] Dolev, D., K. Karplus, A. Siegel, A. Strong, and J. D. Ullman: "Optimal Wiring Between Rectangles"; *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing,* May 1981, pp. 312–317.

[Do79] Donath, W. E.: "Placement and Average Interconnection Lengths of Computer Logic"; *IEEE Transactions on Circuits and Systems,* Vol. CAS-26, No. 4 (April 1979), pp. 272–277.

[ElG81] El-Gamal, A.: "Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits"; *IEEE Transactions on Circuits and Systems,* Vol. CAS-28, No. 2 (February 1981), pp. 127–138.

[ElGSy81] El-Gamal, A., and Z. Syed: "A Stochastic Model for Interconnection in Custom Integrated Circuits"; *IEEE Transactions on Circuits and Systems,* Vol. CAS-28, No. 9 (September 1981), pp. 888-894.

[Ev79] Even, S.: *Graph Algorithms;* Computer Science Press, Potomac, MD, 1979.

[EPL72] Even, S., A. Pnueli, and A. Lempel: "Permutation Graphs and Transitive Graphs"; *Journal of the Association for Computing Machinery,* Vol. 19, No. 3 (July 1972), pp. 400–410.

[GaJo79] Garey, M. R., and D. S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-completeness;* W. H. Freeman & Co., San Francisco, 1979.

[Goo80] Goodhue, E.: *private communication,* February 1981.

[Ha75] Hadlock, F.: "Finding a Minimum Cut of a Planar Graph in Polynomial Time"; *SIAM Journal on Computing,* Vol. 4, No. 3 (September 1975), pp. 221–225.

[HaKu72] Hanan, M., and J. M. Kurtzberg: "Placement Techniques"; in *Design Automation of Digital Systems: Theory and Techniques,* Prentice-Hall, 1972 (M. A. Breuer, *ed.*), Chapter 5.

[HaSt71] Hashimoto, A., and J. Stevens: "Wiring Routing by Optimizing Channel Assignment within Large Apertures"; *Proceedings of the Eighth Design Automation Workshop,* IEEE, 1971, pp. 155–169.

[HMD78] Heller, W. R., W. Mikhail, and W. Donath: "Prediction of Wiring Space Requirements for LSI"; *Journal of Design Automation and Fault Tolerant Computing,* Vol. 2 (1978), pp. 117–144.

[HSM82] Heller, W. R., G. Sorkin, and K. Maling: "The Planar Package Planner for System Designers"; *Proceedings of the Nineteenth Design Automation Conference,* June 1982, pp. 253–260.

[Hi74] Hightower, D.: "The Interconnection Problem: A Tutorial"; *Computer,* Vol. 7, No. 4 (April 1974), pp. 18–32.

[Hs79] Hsueh, M.-Y.: *Symbolic Layout and Compaction of Integrated Circuits;* Memo No. UCB/ERL–M79/80 (Ph.D. dissertation), Electronics Research Laboratory, Univ. of California, Berkeley, December 1979.

[Jo79] Johannsen, D. L.: "Bristle Blocks: a Silicon Compiler;" *Proceedings of the Caltech Conference on VLSI,* January 1979, pp. 303–310. Also appears in the *Proceedings of the Sixteenth Design Automation Conference,* June 1979, pp. 310–313.

[Jo81] Johannsen, D. L.: *Silicon Compilation;* Technical Report No. 4530 (Ph.D. dissertation), Dept. of Computer Science, Caltech, Pasadena, Ca., 1981.

[Ka80] Kajitani, Y.: "On Via Hole Minimization of Routing on a 2-layer Board"; *Proceedings of the IEEE 1980 International Conference on Circuits and Computers,* June 1980, pp. 295–298.

[KaKa79] Kawamoto, T., and Y. Kajitani: "The Minimum Width Routing of a 2-Row 2-Layer Polycell-Layout"; *Proceedings of the Sixteenth Design Automation Conference,* June 1979, pp. 290–296.

[KrvanL82] M. R. Kramer, and J. van Leeuwen: *Wire-Routing is NP-complete;* Technical Report RUU–CS–82–4, University of Utrecht, the Netherlands, February 1982.

[LaP80a] LaPaugh, A. S.: *Algorithms for Integrated Circuit Layout: An Analytic Approach;* MIT/LCS/TR–248 (Ph.D. dissertation), November 1980.

[LaP80b] LaPaugh, A. S.: "A Polynomial–time Algorithm for Optimal Routing Around a Rectangle"; *Proceedings of the Twenty–first Annual IEEE Symposium on Foundations of Computer Science,* October 1980, pp. 282–293;

[La76] Lawler, E. L.: *Combinatorial Optimization: Networks and Matroids;* Holt, Rinehart and Winston, New York, 1976.

[Lee61] Lee, C. Y.: "An Algorithm for Path Connections and Its Applications"; *IRE Transactions on Electronic Computers,* VEC–10, September 1961, pp. 346–365.

[Lei80] Leiserson, C. E.: "Area–Efficient Graph Layouts (for VLSI)"; *Proceedings of the Twenty–first Annual IEEE Symposium on Foundations of Computer Science,* October 1980, pp. 270–281.

[LeiPi81] Leiserson, C. E., and R. Y. Pinter: "Optimal Placement for River Routing"; *Proceedings of the CMU Conference on VLSI Systems and Computations,* October 1981, pp. 126–142. A revised version will appear in *SIAM Journal on Computing.*

[LSV82] Lipton, R. J., R. Sedgewick, and J. Valdes: "Programming Aspects of VLSI"; *Proccedings of the Ninth Annual ACM Symposium on Principles of Programming Languages,* January 1982, pp. 57–65.

[LNSVV82] Lipton, R. J., S. C. North, R. Sedgewick, J. Valdes, and G. Vijayan: "ALI: a Procedural Language to Describe VLSI Layouts"; *Proceedings of the Nineteenth Design Automation Conference,* June 1982, pp. 467–474.

[LloRa81] Lloyd, E. L., and S. S. Ravi: *One Layer Routing Between Parallel Rows of Terminals;* Department of Computer Science Technical Report 81–4, University of Pittsburgh, July 1981.

[MaKu82] Marek-Sadowska,M. and E.S. Kuh: "A New Approach to Channel Routing"; *Proceedings of the 1982 International Symposium on Circuits and Systems,* May 1982, pp. 764–767.

[MeaCo80] Mead, C., and L. Conway: *Introduction to VLSI Systems;* Addison Wesley, Reading, Mass., 1980.

[MiVa80] Micali, S., and Vazirani, V. V.: "An $O(\sqrt{|V|}\cdot|E|)$ Algorithm for Finding Maximum Matching in General Graphs"; *Proceedings of the Twenty-first Annual Symposium on Foundations of Computer Science,* October 1981, pp. 17–27.

[OrDo72] Orlova, G. I., and Y. G. Dorfman: "Finding the Maximum Cut in a Graph"; *Engineering Cybernetics,* Vol. 10 (1972), pp. 502–506.

[PeRu81] Penfield, P. Jr., and J. Rubinstein: "Signal Delay in RC Tree Networks"; *Proceedings of the Second Caltech Conference on VLSI,* January 1981, to appear sometime before 1985.

[PDS77] Persky, G., D. N. Deutch, and D. G. Schweikert: "LTX — A Minicomputer-Based System for Automated LSI Layout"; *Journal of Design Automation and Fault-Tolerant Computing,* Vol. 1, No. 3 (May 1977), pp. 217–255.

[Pi81] Pinter, R. Y.: "Optimal Routing in Rectilinear Channels"; *Proceedings of the CMU Conference on VLSI Systems and Computations,* October 1981, pp. 160–177.

[Pi82a] Pinter, R. Y.: "On Routing Two-Point Nets Across a Channel"; *Proceedings of the Nineteenth Design Automation Conference,* June 1982, pp. 894–902.

[Pi82b] Pinter, R. Y.: "Optimal Layer Assignment for Interconnect"; to appear in the *Proceedings of the IEEE International Conference on Circuits and Computers,* September–October 1982.

[Pr79] Preas, B. T.: *Placement and Routing Algorithms for Hierarchical Integrated Circuit Layout*; Computer Systems Laboratory Technical Report No. 180/SEL–79–032 (Ph.D. dissertation), Stanford University, August 1979.

[Ra79] Raffel, J. I.: "On the Use of Nonvolatile Programmable Links for Restructurable VLSI"; *Proceedings of the Caltech Conference on VLSI,* January 1979, pp. 95–104.

[Ric81] Richards, D.: *Complexity of Single-Layer Routing;* unpublished manuscript, Dept. of Mathematics, Indiana University – Purdue University, July 1981.

[Riv82] Rivest, R. L.: "The 'PI' (Placement and Interconnect) System"; *Proceedings of the Nineteenth Design Automation Conference,* June 1982, pp. 475–481.

[RBM81] Rivest, R. L., A. E. Baratz, and G. L. Miller: "Provably Good Channel Routing Algorithms"; *Proceedings of the CMU Conference on VLSI Systems and Computations,* October 1981, pp. 153–159.

[RiFi82] Rivest, R. L., and C. M. Fiduccia: "A 'Greedy' Channel Router"; *Proceedings of the Nineteenth Design Automation Conference,* June 1982, pp. 418–424.

[Ro81] Rosenberg, A. L.: "Three-Dimensional Integrated Circuitry"; *Proceedings of the CMU Conference on VLSI Systems and Computations,* October 1981, pp. 69–80.

[SaBh80] Sahni, S., and A. Bhatt: "Complexity of the Design Automation Problem"; *Proceedings of the Seventeenth Design Automation Conference,* June 1980, pp. 402–411.

[Sh82] Shrobe, H. E.: "The Data Path Generator"; *Proceedings of the M.I.T. Conference on Advanced Research in VLSI,* January 1982, pp. 175–181.

[Sie81] Siegel, A.: "Fast Optimal Placement for River Routing"; *unpublished manuscript,* Stanford University, November 1981.

[SieDo81] Siegel, A., and D. Dolev: "The Separation for General Single-Layer Wiring Barriers"; *Proceedings of the CMU Conference on VLSI Systems and Computations,* October 1981, pp. 143–152.

[SSC82] Siskind, J. M., J. R. Southard, and K. W. Crouch: "Generating Custom High Performance VLSI Designs from Succinct Algorithmic Descriptions"; *Proceedings of the M.I.T. Conference on Advanced Research in VLSI,* January 1982, pp. 28–40.

[Sou81] Soukup, J.: "Circuit Layout"; *Proceedings of the IEEE,* Vol. 69, No. 10 (October 1981), pp. 1281–1304.

[SouRo81] Soukup, J., and J. Royle: "On Hierarchical Routing"; *Journal of Digital Systems,* Vol. 5, No. 3 (Fall 1981), pp. 265–289.

[StSu79] Steele, G. L. Jr., and G. J. Sussman: "Constraints"; *Proccedings of APL '79* (invited paper); *ACM SIGPLAN STAPL APL Quote Quad,* Vol. 9., No. 4 (June 1979), pp. 208-225.

[Stee80] Steele, G. L. Jr.: *The Definition and Implementation of a Computer Programming Language Based on Constraints;* Ph. D. Dissertation, A. I. Memo 595, Artificial Intelligence Laboratory, M.I.T., December 1980.

[Sto80] Storer, J. A.: "The Node Cost Measure for Embedding Graphs on the Planar Grid"; *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing,* April 1980, pp. 201–210.

[Sz81] Szymanski, T. G.: *Dogleg Channel Routing is NP-complete*; unpublished manuscript, Bell Laboratories, September 1981.

[Ta81] Tarjan, R. E.: "Fast Algorithms for Solving Path Problems"; *Journal of the Association for Computing Machinery,* Vol. 28, No. 3 (August 1981), pp. 594–614.

[Th80] Thompson, C. D.: *A Complexity Theory for VLSI*; Technical Report CMU-CS-80-140 (Ph.D. dissertation), Carnegie-Mellon University, August 1980.

[To80] Tompa, M.: "An optimal solution to a wire-routing problem;" *Proccedings of the Twelfth Annual Symposium on Theory of Computing,* April–May 1980, pp. 161–176.

[Ul70] Ulrich, J. W.: "A Characterization of Planar Oriented Graph"; *SIAM Journal of Apllied Mathematics,* Vol. 18, No. 2 (March 1970), pp. 364–371.

[vanC76] van Cleemput, W. M.: "Mathematical Models for the Circuit Layout Problem"; *IEEE Transactions on Circuits and Systems,* Vol. CAS-23, No. 12 (December 1976), pp. 759-767.

[vanLOtt73] van Lier, M. C., and H. J. M. Otten: "On the Mathematical Formulation of the Wiring Problem"; *International Journal on Circuit Theory and Applications,* Vol. 1, No. 2 (1973), pp. 137–147.

[WieMl82] M. Wiesel, and D. A. Mlynski: "Two-Dimensional Channel Routing and Channel Intersection Problems"; *Proceedings of the Nineteenth Design Automation Conference,* June 1982, pp. 733–739.

[Ye70] Yen, J. Y.: "An Algorithm for Finding Shortest Routes from All Source Nodes to a Given Destination in General Networks;" *Quarterly of Applied Mathematics,* Vol. 27, No. 4, July 1970, pp. 526-530.

[YoKu82] Yoshimura, T., and E. S. Kuh: "Efficient Algorithms for Channel Routing"; *IEEE Transactions on Computer Aided Design for Circuits and Systems,* Vol. 1, No. 1 (January 1982), pp. 25–35.

[Zi81] Zippel, R. E.: *private communication,* July 1981.

Biographical Note

The author was born on December 15, 1953 in Haifa, Israel, where he was raised. He attended the Hebrew Reali Highschool from which he graduated first in the class of 1971. In October 1971 he joined the academic reserve of the Israeli Defense Forces (IDF), and after a basic training period attended the Technion, Israel Institute of Technology in March 1972. He received his B.Sc. in Computer Science *summa cum laude* in March 1975, graduating in three—rather than the usual four--years. Then, in May 1975 he joined the IDF for the regular three-year service in the Communication Corps.

In August 1978 he arrived in the United States to attend the Massachusetts Institute of Technology as a graduate student in the department of Electrical Engineering and Computer Science. He received his S.M. in Computer Science in February 1980 after completing a thesis under the supervision of Prof. Ronald L. Rivest.

He accepted a temporary position as a member of the technical staff with the Computing Sciences Research Center, Bell Laboratories, Murray Hill, NJ, starting September 1982. He will be staying at Bell before going back to Israel during the Fall of 1983.

While at the Technion, he met Shlomit Werner, whom he married in 1976. Their daughter Noa was born in Boston during the January thaw of 1980.

CS-TR Scanning Project
Document Control Form

Date : 7/14/95

Report # LCS-TR-991

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 144 (149-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): 45, 46

Photographs/Tonal Material (by page number):_____

Other (note description/page number):

Description : Page Number:

(A) IMAGE MAP (1-2) UNN'ED TITLE AND ABSTRACT PAGES
(3-144) PAGES #'ED 3-144
(145-146) SCANCONTROL PRINTER'S NOTES
(147-149) TRGTS (3)

(B) MOST PAGES HAVE SMALL BLACK DOTS.

Scanning Agent Signoff:

Date Received: 7/14/95 Date Scanned: 7/26/95 Date Returned: 7/27/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 7/26/1995

M.I.T. Libraries
Document Services