

THE UNIVERSITY OF CALGARY

**An Expert System for Routing VLSI Designs**

by

Mary Margaret Keefe

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

JUNE, 1987

© Mary Margaret Keefe, 1987.

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-38012-8

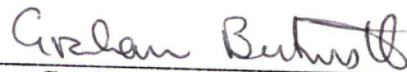
The University Of Calgary

Faculty Of Graduate Studies

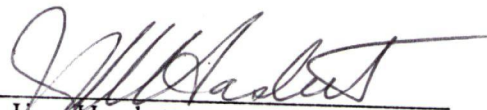
The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled, "An Expert System for Routing VLSI Designs " submitted by Mary Keefe in partial fulfillment of the requirements for the degree of Master of Science.



Supervisor,  
Dr. John Kendall  
Department of Computer Science



Dr. Graham Birtwistle  
Department of Computer Science



Dr. Jim Haslett  
Department of Electrical Engineering

June 22, 1987

## **Abstract**

Given the complexity of VLSI design, a need is created for automated tools that are as competent as human designers at their tasks. Routing, one of these automated tasks, does not yet perform as well as the human designer. Given the inherent complexity of the routing problem a heuristic approach to routing using many and varied constraints is merited. In this thesis, current heuristics for routing switchboxes and channels are investigated to find what heuristics are used on each type of problem, and where they may be improved. Based on the heuristics used to route switchboxes, new heuristics are presented that better estimate the tracks available for routing a net, resolve net conflicts based on this estimate, and allow nets to overlap at the corners of routes to enhance the router's set of heuristics in an effort to achieve 100 per cent routing completion. An explanation is given as to how these heuristics can also be used to route channels. The heuristics are implemented as an expert system using the Automated Reasoning Tool expert system shell. Several difficult switchbox routing problems are solved using the heuristics and produce comparable results to those of current approaches. New heuristics can be added to the modular and flexible expert system to upgrade the router to work within and by the constraints posed by the VLSI design specifications and design environment.

## **Acknowledgements**

Whom should I thank? There are so many! I would like to thank my supervisor, John Kendall, for giving me the freedom to pursue my own endeavours on this thesis. I would like to thank Brian Schack for his help in formatting the figures of this document and for his patience in allowing me to bounce ideas off of him. I would also like to thank my family, friends, and peers for their support, reassurance, and advice, most of which I did not heed. And finally, I thank my mother and father for their emotional support that helped me to push on to the end.

## Table of Contents

Abstract .....	iii
Acknowledgements .....	iv
Table of Contents .....	v
List of Figures .....	viii
List of Tables .....	xi
Chapter 1 Introduction .....	1
1.1 VLSI Routing .....	1
1.2 The Difficulty of the Routing Problem .....	3
1.3 A Heuristic Algorithm for Routing .....	5
1.4 An Expert System for Routing in VLSI .....	8
1.5 Thesis Outline .....	9
Chapter 2 Previous Routing Approaches .....	10
2.1 Models of the Routing Problem .....	10
2.2 The Design Environment .....	11
2.3 Early Routing Approaches .....	14
2.4 Channel Routing .....	16
2.5 Switchbox Routing .....	26

2.6 Other Routing Techniques .....	40
2.7 Discussion of Current Routing Methods .....	42
Chapter 3 New Heuristics for VLSI Routing .....	45
3.1 Background Work .....	45
3.2 New Heuristics for Routing Switchboxes .....	46
3.3 Corner Overlap .....	54
3.4 Heuristics for Channel Routing .....	57
3.5 Summary .....	63
Chapter 4 The B & D Router: An Expert System for Routing VLSI Designs .....	65
4.1 The Use of Expert Systems for Routing .....	65
4.2 The Architecture of the B & D Router .....	69
4.3 B & D Problem Representation .....	70
4.4 The B & D Program .....	73
4.5 Disadvantage of Expert Systems .....	75
4.6 Summary .....	78
Chapter 5 Experiments with the B & D Router .....	79

5.1 Input and Output .....	79
5.2 An Example Run with B & D .....	81
5.3 B & D Against Other Switchbox Routers .....	85
5.3.1 A Simple Switchbox .....	85
5.3.2 A Second Simple Switchbox .....	87
5.3.3 Burstein's Difficult Problem .....	88
5.4 B & D's Comparison Against the WEAVER .....	91
5.5 Summary .....	92
Chapter 6 Conclusions and Future Work .....	93
6.1 Conclusions .....	94
6.2 Future Work .....	95
6.3 Concluding Remarks .....	97
References .....	98



## List of Figures

Figure 2.1 The Floorplans of Three Design Methodologies .....	12
Figure 2.2 Solutions to Grid Alignment Problem .....	14
Figure 2.3 Wave Propagation from X to Y .....	15
Figure 2.4 Line Propagation using X and Y .....	17
Figure 2.5 Line Propagation Dead End .....	17
Figure 2.6 Problem with Horizontal and Vertical Constraints .....	19
Figure 2.7 Column Density Example .....	20
Figure 2.8 Track Filling Example with Overlap .....	22
Figure 2.9 Constrained Left-Edge Solution .....	22
Figure 2.10 Circuit with a Cyclic Constraint .....	24
Figure 2.11 Column Sweep Route of Three Nets .....	25
Figure 2.12 Terminal Constraint on a Switchbox Problem .....	27
Figure 2.13 Loop Area Routing Scheme Example .....	28
Figure 2.14 Pattern Routing Example .....	30
Figure 2.15 Burstein's Solution to his Difficult Switchbox Problem .....	31
Figure 2.16 Hamachi's Solution to Burstein's Problem .....	32
Figure 2.17 Marek-Sadowska's Net Classifications .....	33
Figure 2.18 Constraint Propagation .....	34
Figure 2.19 Expansion Direction Changed by Constraint Propagation .....	35
Figure 2.20 Marek-Sadowska's Solution to Burstein's Problem .....	36
Figure 2.21 Minimum Rectilinear Spanning and Steiner Trees .....	37

Figure 2.22 Joobbani's Solution to Burstein's Problem .....	38
Figure 2.23 Corner and Parallel Overlap .....	39
Figure 2.24 Overlap Dilemma .....	43
Figure 3.1 Available Tracks .....	49
Figure 3.2 Choosing the First Versus the Last Available Track .....	50
Figure 3.3 Example of Net Conflict .....	51
Figure 3.4 Net Conflict Resolution Examples .....	52
Figure 3.5 Net Conflict Resolution Using New Heuristics .....	53
Figure 3.6 Final Solution .....	54
Figure 3.7 Conflict Resulting in a Corner Overlap .....	55
Figure 3.8 Simple Corner Overlap Example .....	56
Figure 3.9 Complicated Corner Overlap Fanout Example .....	58
Figure 3.10 Channel Routing Problem with no Cyclic Constraint .....	60
Figure 3.11 Channel Routing Problem with Cyclic Constraint .....	61
Figure 3.12 Channel Routing Problem Requiring Net Merging .....	62
Figure 4.1 Channel Schema Definitions .....	70
Figure 4.2 Row and Column Relations .....	71
Figure 4.3 Pin Schema Definitions .....	71
Figure 4.4 Net Schema Definition .....	72
Figure 4.5 Connection Schema Definition .....	72
Figure 4.6 Example ART Rule .....	76
Figure 4.7 Procedural Versus ART Code .....	78
Figure 5.1 B & D Input Example .....	80
Figure 5.2 B & D Example Output .....	80
Figure 5.3 First Step of an Example Switchbox Routing Problem	

.....	81
Figure 5.4 Step Two .....	82
Figure 5.5 Step Three .....	83
Figure 5.6 Step Four .....	84
Figure 5.7 Solutions to a Simple Switchbox Problem .....	86
Figure 5.8 Solutions to a Second Simple Switchbox Problem .....	87
Figure 5.9 Luk's Solution to Burstein's Problem .....	89
Figure 5.10 B & D's Solution to Burstein's Problem .....	90

## **List of Tables**

Table 4.1 Summary of the Rules Written for B & D .....	74
Table 5.1 Statistics for the Simple Switchbox in Figure 5.7 .....	86
Table 5.2 Statistics for the Second Simple Switchbox .....	88
Table 5.3 Statistics for Burstein's Difficult Switchbox .....	90
Table 5.4 Execution Statistics of B & D and the WEAVER .....	91

# CHAPTER 1

## Introduction

The advent of the integrated circuit industry in the late 1960's revolutionized the number and types of tasks that could be performed by a computer. It also created a new branch of software development in the area of computer-aided design (CAD) for performing tasks involved in chip design. In the past, designers were able to lay out SSI (Small Scale Integrated) and MSI (Medium Scale Integrated) circuits quickly and efficiently, because the designs required the layout of small numbers of components that could be managed and designed by hand. Matters are somewhat more complex with LSI (Large Scale Integrated) circuits, but still manageable. Technological improvements have now allowed designers to build smaller, faster, and subsequently more complex circuits on a single chip. Chip designers now place upwards of 10,000 to 100,000 transistors together to create a VLSI (very large scale integrated) chip. These designs require the use of CAD tools to develop the entire chip from its initial top level functional specification down to the layout of the components on the chip.

### 1.1. VLSI Routing

In the VLSI design process, a variety of tasks are performed that lay out cell component specifications, check for design rule violations, place cells in fixed positions on the design following the topology of the interconnections that have to be made between cells and the physical geometry of the chip. The concentration of this thesis is on routing, a task performed late in the VLSI design process. Given a chip definition and a placement of cells (the functional blocks) on the chip, the routing problem

is to optimally route within the physical constraints of the chip 100 per cent of all connections such that no two distinct connections intersect so as to become equipotential. The chip boundaries and cell placement, which includes input/output pads, determine where and how much routing area is available. Other physical constraints include the positions of terminals on each cell's boundaries that in turn will connect to terminals on the boundaries of other cells. Each terminal has a fixed location on the chip and is assigned a net name for interconnection. A net consists of a set of terminals which are to be connected -- to share the same signal and thus be equipotential. A path through the available routing areas on the chip is established for each net and is laid down as wires on the chip according to design rule specifications of the technology used. Two layers are commonly used for routing; nets may cross over one another if they are routed in different layers. Vias (contact cuts) are used to change layers. If different nets should cross each other's paths on the same layer, they will share the same signal, and thus become equipotential.

Once defined, routing looks to be a simple geometrical problem, in practice, it is a labour intensive task. VLSI designs on the order of 10,000 nets may take a designer three to four months to route by hand. In some instances, 50 per cent of the design time [Souk81] and 80 per cent of the chip area may be taken up by routing [Mead80] [Sche86]. Automating routing is attractive because of the speed up in design time; automated routers can route hundreds of nets in a few minutes as opposed to a few months by hand. But automated systems have had difficulty obtaining the same quality of routing that designers have which is usually measured by how compact the routing solution is. It is well known that results obtained by automatic routers are not as compact as those obtained by human designers especially on custom chips which have a more irregular and

denser layout than chips designed using other methodologies [Aven83]. *A task to which so much design time and area is devoted deserves to be performed properly.* The next section discusses why routing is a difficult and time consuming task and why heuristic algorithms can be used solve it.

## 1.2. The Difficulty of the Routing Problem

No definition of 'optimal' routing really exists. For some extremely difficult routing problems, 100% interconnection of all nets cannot be guaranteed (see Section 2.5; Switchbox Routing). Most designers classify good routing as those solutions which minimize area, while trying to complete all connections. Solutions which keep the chip area minimized can keep the chip yield high. However, because wires are the objects that are actually routed and not 'area', the routing problem must be translated from minimizing chip area -- a two dimensional minimization -- to that of minimizing wire length, minimizing the number of vias, or some other one dimensional phenomena that routers deal with. Indeed, two early routers minimized wire length in an attempt to achieve good routing [Lee61] [High69], but because the constraint of wire length disregards many other constraints that affect the area of routing -- such as the density of interconnect, an important factor which directly affects the attainment of 100% routing completion -- incompleted solutions or solutions whose quality was unacceptable were obtained.

From a different perspective, the routing problem thought of as a 'random problem' according to Abu-Mostafa's definition [Abu87]. A random problem lacks sufficient structure in its definition that an algorithm cannot be stated for it in mathematical terms. Instead, large amounts of detailed information of the possible cases of the problem must be kept for comparison to later versions of the problem. He cites as an

example of a random problem, the problem of recognizing patterns in a natural environment. The pattern he selected to recognize was a tree. Considering all the possible cases of trees, their branches, leaves, and other properties, there is sufficient randomness in the problem of tree recognition that a simple algorithm cannot possibly recognize all trees. Because the algorithm is general, it recognizes at best a small subset of trees and most likely include tree-like objects which are in fact not trees. The routing problem is the same in this respect and is random because its definition cannot exactly specify the optimal routing solution for all possible routing problems. At best it can use simple algorithms to come close to the optimal. Abu-Mostafa's solution to his problem was to memorize all possible combinations of trees and then match the object to a tree in memory, an impossible feat to perform since the number of combinations is possibly limitless.

The routing problem is also classified as belonging to the NP (Non-deterministic Polynomial) class of problems [Aho74] [Szym85] if a definition of optimal exists, no algorithm can be specified that will optimally solve this class of problem in polynomial time [Baas78].<sup>1</sup> *To generate all possible solutions and search sequentially through them to find the optimal one would take exponential time.* One alternative to this procedure is to expand the original problem into a search tree with branches leading to many partial solutions and then pick the best local path to continue with. However, it is a characteristic of NP problems, that following one branch of a search tree to a partial solution in no way gives information on how much better or worse another path of the tree is unless a significant number of paths (nearly all) are expanded [Mead80].

---

<sup>1</sup>Algorithms with an exponential bound have time complexities of the order  $X^n$ , where  $n$  is the number of inputs to the problem. Algorithms with a polynomial bound are of the order  $N^X$ .  $X$  in both cases can represent the number of times all  $n$  inputs are considered by the algorithm or the number of variables by which each of  $n$  is evaluated.



The most common method of solving NP problems is to develop heuristic algorithms which embody 'rules of thumb'. Using information about the current problem state, the appropriate heuristic is applied to change the current problem state to a new state, leading the partial solution on the path towards a 'good' final routing solution. An optimal solution cannot be guaranteed using heuristics, but theoretically, close to optimal can be [Baas78].

To sum up both theories, an optimal solution to the routing problem can only be obtained by gathering a significant amount of information for comparison, whether it be patterns in memory or an exponential number of generated solutions. An alternative way to proceed is to develop heuristic algorithms to solve the routing problem.

### **1.3. A Heuristic Algorithm for Routing**

The purpose of a heuristic routing algorithm is to define how to solve the routing problem accurately using heuristics. To do this the algorithm must define a model of constraints that represents the state of the problem and define a set of heuristics that will find a solution that is close to the optimal in all cases where that solution exists. Given this task, the algorithm can only fail in two instances. It can fail if the model does not represent the problem correctly or if the heuristics do not route correctly with respect to the routing definition.

If the routing definition is referred to strictly as a set of physical constraints which the algorithm must meet, then it can be said that the model will only fail to represent the routing problem correctly if it either ignores necessary physical constraints or imposes unnecessary ones with respect to the routing definition. For example, a model which ignores the boundaries of the routing area would allow routes to wander outside the

boundary of the channel which violates the routing definition. In the same manner, imposing the use of the two-direction-two-layer wiring model is an unnecessary physical constraint. The routing definition states that nets cannot intersect such that they become equipotential. Nets have to be routed on different layers when they cross, but they do not have to be routed on different layers in different directions. This restriction is used, of course, because it simplifies the routing problem; the routing method no longer requires a set of heuristics to make decisions on which layer a net will be routed. However, it can prevent the router from finding a good solution.

Heuristics can also be viewed as trying to meet the physical constraints of the routing definition, specifically the constraints of laying down routes that are in a 'close to optimal' configuration. Thus similar to modeling the routing problem, heuristics can also ignore necessary physical constraints. For example, an algorithm minimizing wire length routes well in situations where there are no net crossovers in the solution. When crossovers exist, one routed net can block another net from completion. In this situation, nets which pass through the densest areas of the routing region are given priority over others, because it ensures that they will not be blocked later. From a more perverse perspective, heuristics can also be thought of as imposing unnecessary physical restrictions on routing solutions. By their own definition heuristics are local optimizations and do not usually find the absolute optimal solution to the routing problem. Indeed they may by chance restrict a solution from being the optimal merely by their method of solving the problem. They impose unnecessary restrictions which can be overcome by using more heuristics, albeit with their own restrictions. The increasing number of heuristics become less and less restrictive, less simplistic in their

representation of the routing problem. Their combined complex definition of an algorithm is more capable of solving the complexities inherent in the routing problem than any single simplistic restrictive heuristic. Thus adding a heuristic not only adds new criteria for accurately defining a routing solution but also removes a previous restriction that was present without its use.

It is not the purpose of a routing algorithm to find *the* exact definition of the algorithm. Problems with a certain amount of randomness, which can be said to be true of the routing problem, require a full exposition of all possible problems and their solutions to be accurate with respect to the definition, an impossible task which can only be approximated by a handful of heuristics and is better estimated by many increasingly accurate heuristics. The fact that an exact definition has not been ascertained as yet should also emphasize that all but one routing algorithm falls short of the definition and can be improved. This is true for all current routing approaches. In fact, current heuristic algorithms for routing are forced to create generalized heuristics to take care of cases where no detailed heuristics are present to route them. Thus current approaches should be evaluated to find out what heuristics they use to define good routing, where they impose unnecessary restrictions on the solutions to the problem, and what additional criteria are required to improve the quality of routing.

This idea can explain why humans may do well at solving routing problems. When confronted with the routing problem, the designer can analyze the many physical features present, capitalize on her own experience of what should be routed first and last, and route whole or partial nets in any direction and layer she desires using rules of thumb she has learned over the years. It is no coincidence that the more experience a

designer has, the better she is at producing a good routing solution. Experience may mean that she has acquired (1), the knowledge of what heuristics lead to good routing and (2), the knowledge of *when* to apply the heuristics to solve the problem in the best way.

#### **1.4. An Expert System for Routing in VLSI**

The main work of this thesis encompasses the idea of enhancing current approaches to routing by adding more heuristics. Specifically, switchbox and channel routers are examined for limits in their approach to the routing problem. The switchbox and channel problems are forms of the general routing problem which define routing regions to be rectangular. Among the heuristics used to route switchboxes a shortcoming is found in the estimation of what areas of the switchbox are available to nets. This shortcoming jeopardizes the ability of the router to complete 100 per cent of the connections. New heuristics are developed that enhance the current base of switchbox routing heuristics by making a better estimate of the areas that are available for nets to complete their routes.

The secondary work of this thesis involves establishing a set of heuristics that may jointly route the switchbox and channel routing problems. Currently different routing heuristics are used to route each type of problem, because they differ in their models. Channels allow terminals to be defined on two opposite sides of the rectangular routing region whereas switchboxes can have terminals on all 4 sides. Despite their differences, which are elaborated on in chapter 2, the channel model could be defined as a subset of the switchbox and as such should be able to be routed by the same set of heuristics. This possibility is investigated.

The final work of this thesis involves the implementation of these new heuristics in an expert system for routing. Expert systems are a natural way for implementing a heuristic algorithm as they autonomously apply heuristics based on the current state of the problem, as opposed to a conventional program which applies heuristics in coded sequence. As well, expert systems offer a modular and flexible way to develop and fine tune large numbers of heuristics for a difficult problem such as routing. The implementation of the new heuristics using this medium and the expert system shell's performance as a system for developing software is examined in this thesis. [Keef86] describes an earlier exploration of this idea.

### **1.5. Thesis Outline**

Chapter 2 of this thesis explores current routing methods. It analyzes the algorithms with respect to the general heuristics used to route them. The similarities and differences between the heuristics used to route channels and switchboxes, and the differences between methods for resolving conflicts between nets in the switchbox routing problem are investigated. Chapter 3 presents new heuristics which enhance current switchbox heuristics for net expansion, and can also be used to route channels. Chapter 4 discusses the implementation of the B & D router, an expert system for routing in VLSI that encompasses the heuristics discussed in chapters 3. It discusses the overall architecture of the B & D router and the data structures used to represent the routing problem state. Chapter 5 gives an example of B & D's routing capability and compares the B & D router's switchbox routing results to other systems' results. Chapter 6 concludes the thesis and describes the future direction of work regarding routing and expert systems.

## **CHAPTER 2**

### **Previous Routing Approaches**

Previous routing approaches are discussed with the intent of discovering what heuristics and methods have been used to solve them. The models used to represent the general routing problem are presented and the influence from the design environment is discussed. Area routers, channel routers, and switchbox routers, the heuristics and the methods used by each type of router are included in the presentation of previous routing methods. Chapter 2 concludes with a discussion of the three problems facing switchbox routing and the general routing problem.

#### **2.1. Models of the Routing Problem**

A wiring model decides in what directions and layers wires can route. In general, wiring models assume that only two layers are available. Different geometries are available in which to route wires; euclidean, where straight wires travel in any direction, rectilinear or manhattan, where wires travel horizontally or vertically, and boston, where wires can also travel at a 45 degree angle from the vertical and horizontal directions. Most routing algorithms use the two-layer-two-direction wiring model which uses rectilinear geometry and allow one layer to route horizontally and the second layer to route vertically. A contact cut (via) is used to change layers and hence directions.

In addition to the two-layer-two-direction wiring model, automatic routers can use the grid approach to routing. A grid is a lattice of equidistant horizontal and vertical lines mapped onto the routing region. A grid point is formed at the intersection of two perpendicular lines. All terminals on the boundary of the routing region and contact cuts must be

located on grid points and all wires must be routed on grid segments. Grid lines are usually assumed to be separated by the minimum distance required between a contact cut and the routing layer in each direction to avoid design rule violations. Routers that do not use a grid allow terminals to locate anywhere along the boundary of the routing region and their wires to route according to design rule specifications.

## 2.2. The Design Environment

When an automatic router is developed for a CAD system, it takes account of how the design methodology affects the general routing model of the routing problem. The design methodology determines the floorplan of the chip, the location and size of the routing areas, and the layers that are available for routing. Three design methodologies, standard cell (or polycell), gate array, and structured design (or hierarchical design), are discussed below.

A standard-cell floorplan typically has rows of cells interwoven with rows of interconnect space (see Figure 2.1a) [Aven83]. Cells are designed to abut horizontally by being the same height, but they may differ in width. Vertical routing space at the ends of the rows and special route-through cells [Breu83] give routes access to adjacent interconnect rows. The amount of space allotted for interconnection can be altered by the designer. The fabrication process determines the number of layers available for routing, which is typically two.

Gate array design methodology is more restrictive than standard cell design because the layout has been pre-masked. The highly regular, structured, and well-spaced component layout in gate array semi-custom design gives the automatic routers an advantage over those that have to route the tightly packed and irregular layout of full custom designs

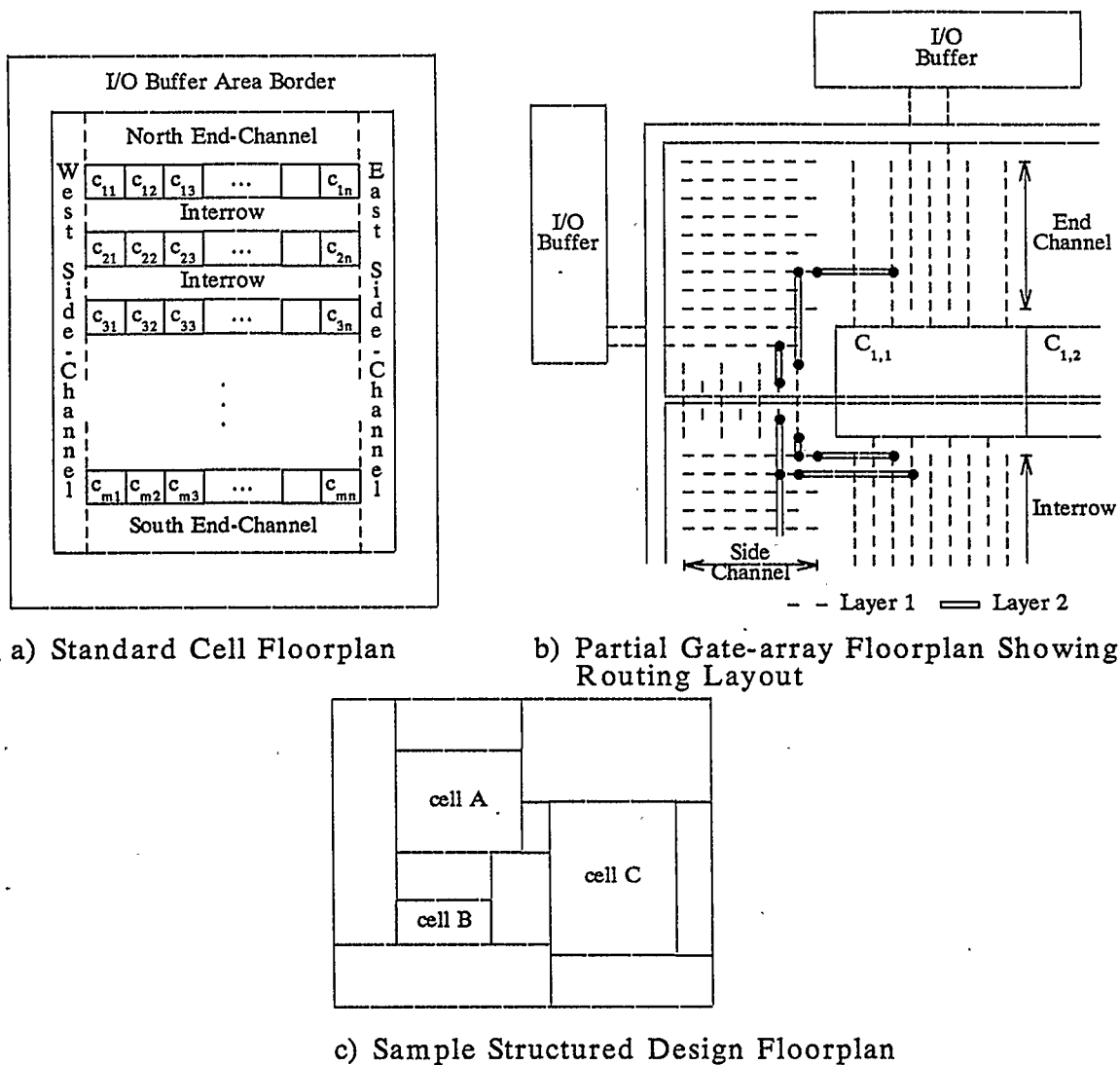


Figure 2.1 The Floorplans of Three Design Methodologies

[Aven83]. However, gate array routers find it harder to achieve 100% routing completion [Jenn84], because (1), the routing areas are set in size



and cannot be expanded if the router requires more space and (2), if all routings are not completed, a designer must weave the leftover routes by hand, which is a difficult and sometimes impossible task owing to the scarcity of available routing space after the automatic router has run.

The most flexible design methodology of the three is structured design [Mead80] and is used to develop full-custom designs. An example floorplan is shown in Figure 2.1c. The position and size of the routing areas are usually set after the cells have been placed on the chip, but this is variable if the routing areas have to be increased from the demands of the router. The choice of layers is decided by the fabrication process. Cells are usually designed to encompass most of the local functional logic they require, so there is less interconnect between cells and therefore less routing space required. Automatic routing on full-custom design does not achieve as tightly packed results as hand-packed routing, because the routing layout is highly irregular.

Another difficulty that full custom designs have is matching up a non grid-based cell design to a grid-based router. Two possible solutions have been proposed by Ousterhout [Oust84]. The first uses a sidewalk boundary around the cells and routes connections within the boundary to the master grid connections (see Figure 2.2a). The second uses a flexible grid approach (see Figure 2.2b), where small areas in the channel use different grid lengths and compose to form the larger, non-uniform grid.

In general, automatic routing methods are developed to solve the general routing problem, but once they are to be applied to a CAD design environment, they are specialized to deal with the restrictions which the design methodology, chip layout, and fabrication process give them.

The following sections discuss the current methods that solve the routing problem. These methods solve the general routing problem, and

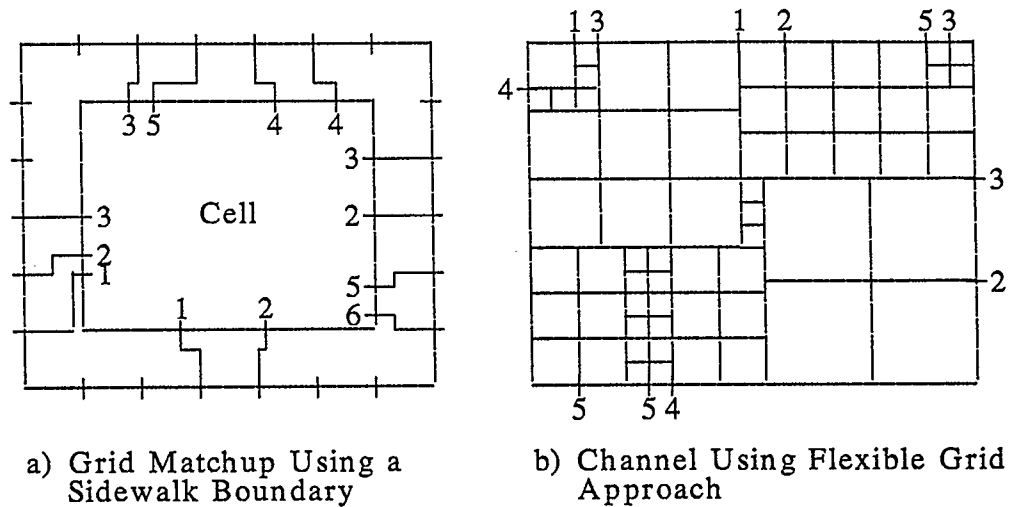


Figure 2.2 Solutions to Grid Alignment Problem

are not specifically directed for use by any design methodology. The discussion emphasizes the heuristics that each router uses to define good routing, the specific improvements each router has made over its predecessors, and the shortcomings each router has.

### 2.3. Early Routing Approaches

The first routing approaches are called area routers. They route one net at a time across the entire chip area. Their main objective is to find paths for nets around the cell and other obstacles in the chip area. Two well-known area routers are the wave propagation and line propagation routing algorithms.

Wave propagation, a derivation of Dijkstra's shortest path algorithm [Aho74] [Lee61], is also known as the Lee-Moore algorithm, the maze runner, and the path finding algorithm [Aker72] [Rubi74]. The method gets its name from the wave that propagates from one terminal of a net across a grid of positions to find the other terminal. Each successive iteration of the wave is labeled with a numeral. When the other terminal

is found, a path is established by following the numbers in reverse order to the source terminal. An example propagation is shown below; X is the source of the wave and Y is the target. The steps of the wave are shown in numerals.

The wave propagation method always finds a path for a net if one exists, which is one of the reasons why wave propagation is used in many CAD systems today, even though it was developed almost thirty years ago. However, wave propagation has a high computational cost because its search area grows exponentially at each iteration. The search area can be reduced by restricting the wave's movement towards the terminals only, or by having waves propagate from all terminals simultaneously. A hardware solution proposed by [Hong83] dedicates processing elements for each source node so that wave propagation is done in parallel for all nets. For serial computation, however, the expansion cost is the main limitation of the wave propagation method.

Line propagation improves on wave propagation by limiting the amount of search made for a possible route for a net [High69]. The method is also known as the Hightower algorithm, the Aim algorithm, and the direct routing method [Souk81]. Each terminal of a net propagates

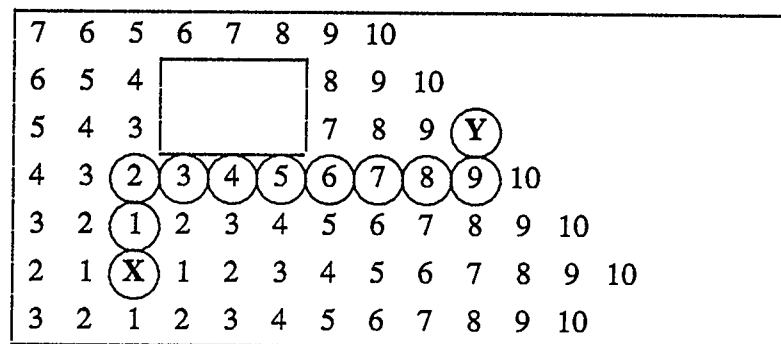


Figure 2.3 Wave Propagation from X to Y

two perpendicular lines that must intersect to establish a completed path. If a line encounters an obstacle, an escape point is formed. A route is established to the escape point and new lines expand from that point. In the example below, escape points of lines a, b, c, and d are formed as the algorithm proceeds to connect X and Y (see figure 2.4).

The computational cost of the line propagation method is lower than that of the wave propagation method, because less positions are being searched; however, solutions are not always found. For example, in Figure 2.5 line b propagates into a box to find a connection to point Y without success and no track is available for a line to propagate back out of the box. This method shows little intelligence for choosing escape routes and is worse for complicated mazes where many obstacles exist.

The area routers discussed above are adequate for small routing problems, but as the density of the routing increases, the heuristic of minimum net length fails to route to 100 per cent completion. Entire nets are routed one at a time, and it is a characteristic of this heuristic that nets which are routed earlier may block later nets. Soukup [Souk81] solves this problem by using flexible routed nets. If a routed net blocks a net from routing later on, the first route can be removed in favour of the second. Once the second net is routed, the first net can proceed to find an alternate route using the wave propagation technique. But minimum net length or the order in which nets are routed cannot decide when one net should have preference over another.

## 2.4. Channel Routing

Area routers have two drawbacks that disallow their use for solving routing problems that contain on the order of 10,000 nets. The size of the problem prevents a large percentage of routes from being completed

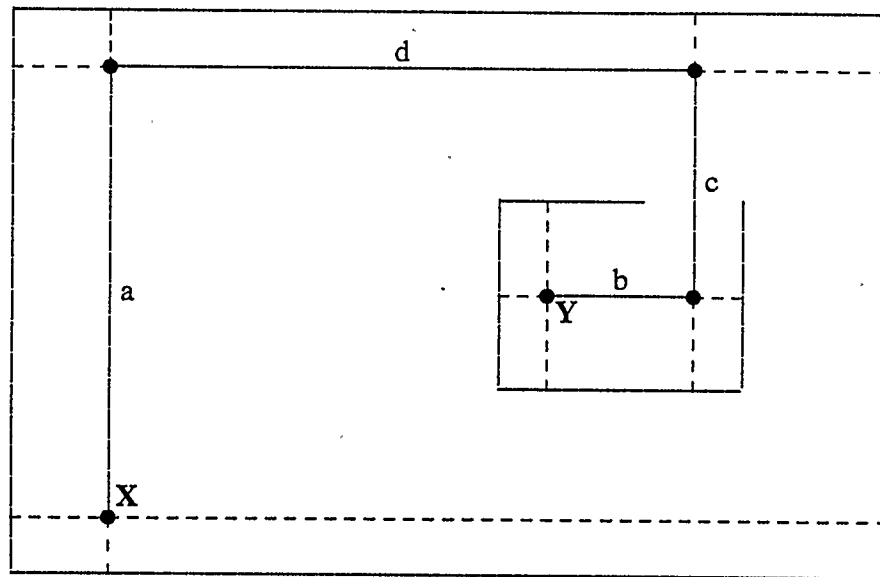


Figure 2.4 Line Propagation using X and Y

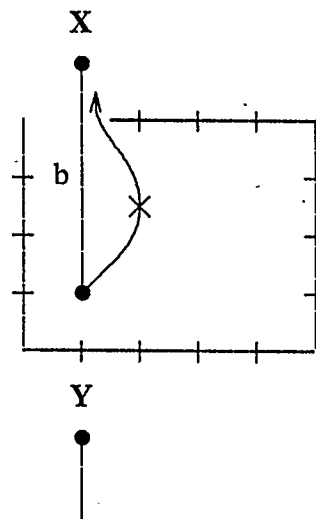


Figure 2.5 Line Propagation Dead End

because they are routed a net at a time and minimizes net length only. And as the density of the interconnect increases, so does the percentage of nets left unrouted. But because the routing problem is NP complete, each

additional constraint beyond that for minimizing net length pushes the time complexity closer to its exponential time bound [Souk79].

An alternative to routing the entire chip area at once is to divide the area into channels, establish general paths for nets through the channels (global routing)<sup>1</sup>, establish the boundary constraints between adjacent channels (net ordering), and perform detailed routing on the channels independently of one another using a specialized router called a channel router.

Formally defined, a channel is a rectangle of routing space that has fixed terminals on two opposing boundaries only. Horizontal channels have terminals on the top and bottom sides; vertical channels have them on the left and right sides. Channels can be defined to be L, T, and X-shaped [Pint81], but this is rarely done. Channels have no obstacles save pre-routed nets, which are nets that have been routed manually.

To spare the reader undue confusion in explaining the subsequent terminology regarding horizontal and vertical channels, the term 'channel' will refer to a horizontal channel and the discussion of channels will refer to horizontal channels.

A column is a vertical line that spans the height of a channel. A column is defined for each terminal or grid point on the upper and lower sides of the channel. Tracks are a horizontal lines which span the length of the channel, but are added to the solution as they are required by the channel router. The columns and tracks are mapped onto a grid upon which nets are routed. Typically, the two-direction-two-layer wiring model is used along with the grid approach. Nets route one vertical

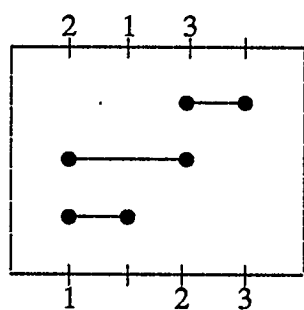
---

<sup>1</sup>Global routers are not discussed in this thesis, because they do not perform detailed routing; they assign nets to channels based on the area available for routing throughout the whole chip. The generally accepted procedure is based on the wave propagation algorithm and can be found in [Souk79].

segment at each terminal position and one horizontal segment to connect the vertical segments providing that vertical and horizontal constraints between all nets are obeyed.

A horizontal constraint exists between two nets if one net starts its horizontal segment before the other has finished. In Figure 2.6a, net 2 has a horizontal constraint with net 1 and net 3.

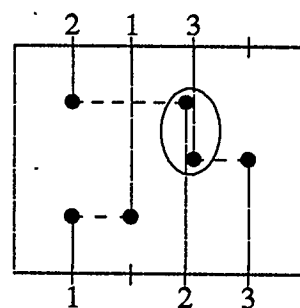
A vertical constraint exists between two nets if both have terminals located on the same column. Vertical constraints are captured in graphs where nodes represent nets. Further, arcs and the position of nodes relative to one another represent the 'above' or 'below' relation. The vertical constraint graph for Figure 2.6a is shown in Figure 2.6b, and shows that net 3 is constrained to route its horizontal segment above net 2's, otherwise they will overlap and become equipotential (see Figure



a) Horizontal Constraints



b) Vertical Constraint Graph



c) Vertical Constraint Violation Causes Overlap

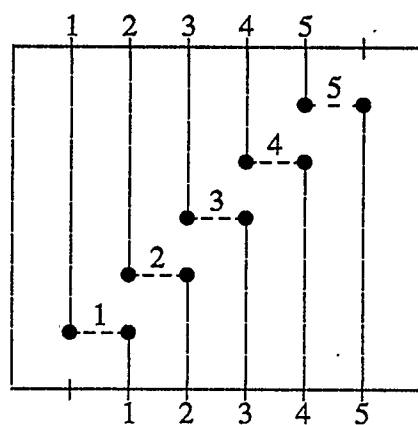
An explanation may be required at this point as to what the information in this figure and the figures that follow represents. A bullet (shaded circle) represents a contact cut. A line represents a segment of a routed net and is bounded by a terminal and a bullet, or two bullets. Dotted and solid lines represent the two different layers used in routing. Any other significant points in the figures will be presented where appropriate.

Figure 2.6 Problem with Horizontal and Vertical Constraints

2.6c).

Because the number of tracks a channel requires is established during routing, an estimate can be made from calculating the channel's density. Channel density is defined to be the maximum of all column densities in a given channel. A column's density is equal to the number of nets that cross it. For example, the channel density in Figure 2.7a is two. However, it takes five tracks to route the example, indicating that channel density does not always give a correct estimate. A better estimate in this case is given by the number of linked nodes in the vertical constraint graph which is five (see Figure 2.7b).

Similar to the constraint of net length in area routing, channel density, vertical and horizontal constraints, and the number of tracks are the constraints by which the horizontal net segments are routed. The terminal pins decide where the vertical segments will go automatically. These constraints and others that are used for channel routing are discussed in the following sections.



a) Routed Example



b) Vertical Constraint Graph

Figure 2.7 Column Density Example

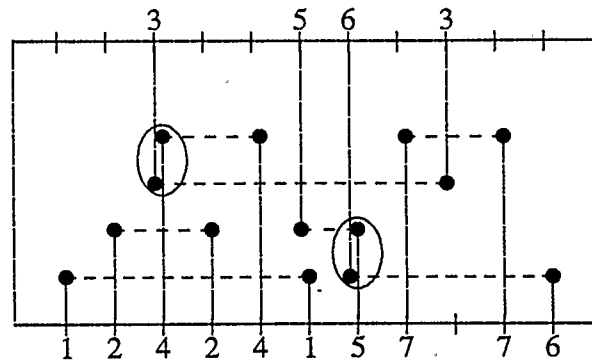


River routing [Mead80] [Pint81], the simplest of all channel routing methods, routes in one layer only, thus nets cannot cross each other. Channel density calculates the theoretical minimum channel width required to adequately route the problem [Ullm84]. Because of its simplicity, river routing has been shown to be optimally solvable in polynomial time [Leis81]. However, it cannot be used to route the general two-layer channel routing problem.

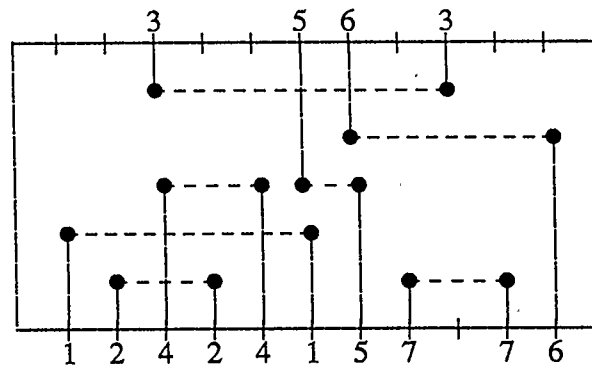
The wave propagation and line propagation can be used for the two-layer channel routing problem. But again they fail to route 100 per cent of all connections because they route a net at a time and disregard all other constraints.

Channel density is the main optimizing constraint in track filling (or left-edge) algorithms for channel routing [Pers78] [Yosh82]. The method proceeds from the bottom-left of the channel across a track filling it with as many horizontal segments of nets as possible while obeying the horizontal constraints of the nets thereby minimizing channel density. However, the method does not follow vertical constraints and therefore some solutions may contain overlaps (see Figure 2.8).

The constrained left-edge algorithm [Mukh86] obeys horizontal and vertical constraints and minimizes channel density by executing a modified track filling procedure. Nets which are upper leaves in a vertical constraint graph route on the top track of a channel; nets which are lower leaves route on the bottom track. The algorithm proceeds left to right filling the tracks according to horizontal constraints. Once the nets are routed, their nodes are deleted from the vertical constraint graph and the next two tracks are filled. This algorithm does not create solutions with overlaps; however, nets are selected on a first come first serve basis based on their location along the channel boundary, which does not guarantee



a) Left-edge Algorithm Causes Overlaps



b) Constrained Left-edge Solution With No Overlaps

Figure 2.8 Track Filling Example with Overlap

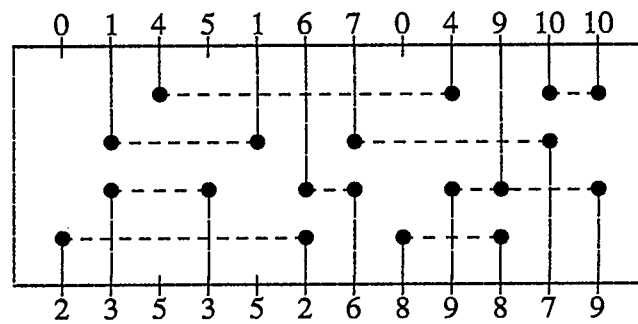


Figure 2.9 Constrained Left-Edge Solution

that solutions minimize channel density.

The least-cost path algorithm [Mukh86] recognizes the relationship between channel density, vertical horizontal constraint graphs, and net merging by filling tracks with nets that are the most 'density reducing'. If a combination of nets merged on one track covers all columns that have maximum channel density, then that combination is defined as density reducing. The algorithm proceeds to build two graphs, one of the upper leaf nodes in the vertical constraint graph, the other of the lower leaves. An arc is established between two nodes in the graph if the two nets can be merged onto the same track. Each arc is assigned a cost equal to the number of columns left uncovered between the two nets that have maximum channel density. The algorithm guarantees that the least cost path through each graph will be found if a path exists. The nets in the selected path from the upper graph are routed on the upper track of the channel. The same is done for the lower graph. Upon the next iteration, two new graphs are built for the leftover nets and the resulting least cost paths are routed. The one restriction to using this algorithm is that the vertical constraint graph must be acyclic.

A cyclic constraint exists between two nets if both enter and exit the channel at the same columns, but at each time on opposite sides of the channel. Problems with a cyclic constraint cannot be solved if nets are allowed to route using only one horizontal segment (see Figure 2.10a). Deutsch's dog-leg router breaks the cycle by allowing nets to route on more than one horizontal segment [Deut76]. A 'dog-leg' is an extra vertical segment placed in a route to give it a knee-bend. An example solution to the cyclic constraint problem using a dog-leg is shown Figure 2.10b. Dog-legs ensure that a cyclic constraint can be routed if the extra vertical columns are available, but require extra contact cuts and thus increase the capacitance and the area of the channel. To limit the number

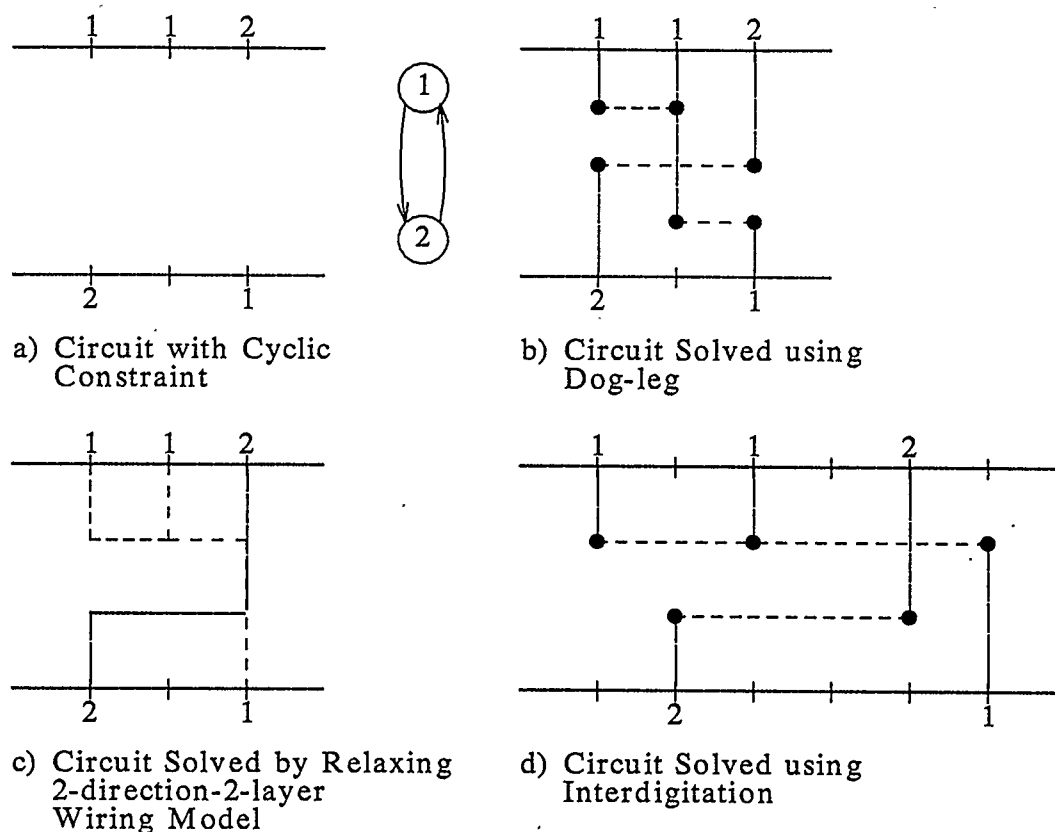


Figure 2.10 Circuit with a Cyclic Constraint

of dog-legs a route can have, Deutsch suggests that dog-legs be introduced at a column where a terminal already exists for the net; however, this restriction cannot be followed between two terminal nets.

The cyclic constraint can also be solved by relaxing the two-layer-two-direction wiring model, as shown in Figure 2.10c; however, parallel overlap is introduced as well which can increase capacitance between the wires in question. Interdigitation [Jenn84], which permits one terminal entry per column, removes cyclic constraints altogether; however, this restriction puts valuable routing area to waste (see Figure 2.10d). All three techniques have increased the channel routing area and capacitance

both of which are necessary to solve the cyclic constraint.

A method called the column sweep approach departs from the track filling method and routes all tracks at once column by column. [Rive81] [Rive82]. The method, also referred to as the greedy algorithm, forces nets to take the closest track to their next exit point when they enter the channel. An example run of a column sweep is shown below in Figure 2.11. Because the channel is swept from left to right, the algorithm cannot determine if a net routed early on will block a net routed later, a similar problem that line and wave propagation algorithms encounter.

The topological approach to routing differs from the other channel routers discussed above because its one heuristic minimizes vias [Mare84]. The nets are topologically arranged on two planar graphs, each graph representing a different routing layer. A via in a net represents a change of layer and therefore a change in graphs. The object is to embed a net one at a time into the graphs using the minimum number of vias necessary to keep each graph planar. Geometric parameters such as channel size and

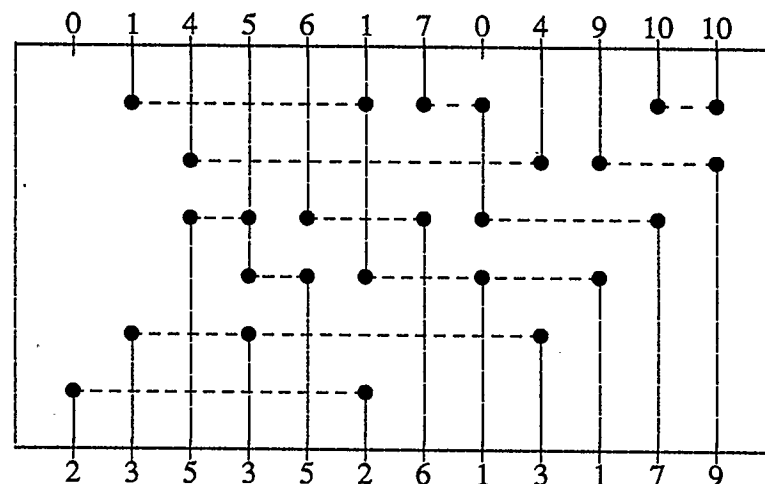


Figure 2.11 Column Sweep Route of Three Nets

capacitance are not considered with the result that some topologically sound solutions cannot be physically realized.

In summary, channels are routing areas containing no obstacles whose height can be expanded during routing if necessary. Different heuristics and techniques are available to find good channel routing solutions. In general, channel routers can use vertical and horizontal constraints, net merging, channel density, and the number of vias to obtain solutions. Good results have been achieved by the least cost path router, which uses three of these heuristics to determine the maximum density reducing routes provided the problem has no cyclic constraint. The solutions to the cyclic constraint either relax the constraint of the two-direction-two-layer wiring model, allow the nets to route using more than one horizontal segment, or restrict columns to having only one terminal assignment. Each technique entails a necessary increase in the area or capacitance in the routing solution because of the extra vias or parallel routing that is introduced.

## **2.5. Switchbox Routing**

A difficult routing problem which has been tackled only recently by automated routing methods is the switchbox routing problem. The switchbox model is similar to the channel except that terminals can be located on all four sides of the switchbox. Their definition provides greater flexibility than that of the channel [Jenn84], because other routing region shapes, such as L, T, and X-shapes can be defined using switchboxes. This problem frequently appears in VLSI design where either the floorplan of the chip or the design procedure sets both the height and width of the routing regions and fixes the locations of the terminals on the switchbox. The restrictions make switchboxes more difficult to route than channels and makes many channel routers incapable

of routing the switchbox problem. First, 100 per cent cannot be guaranteed because the switchbox area cannot be expanded during routing [Souk81]. Thus, track filling channel routers cannot be used to route switchboxes, because they require that tracks can be added while routing is being performed. This problem can be circumvented if the design procedure allows cells to be re-placed after routing to provide more routing space if it is required. Second, the locations of fixed terminals on the four sides of the switchbox constrain where vertical and horizontal segments can route (see Figure 2.12). Track filling routers assume that the horizontal segment of a route can be placed on any track, provided that vertical and horizontal constraints are followed. Thus switchbox routers cannot follow the same procedure as channel routers.

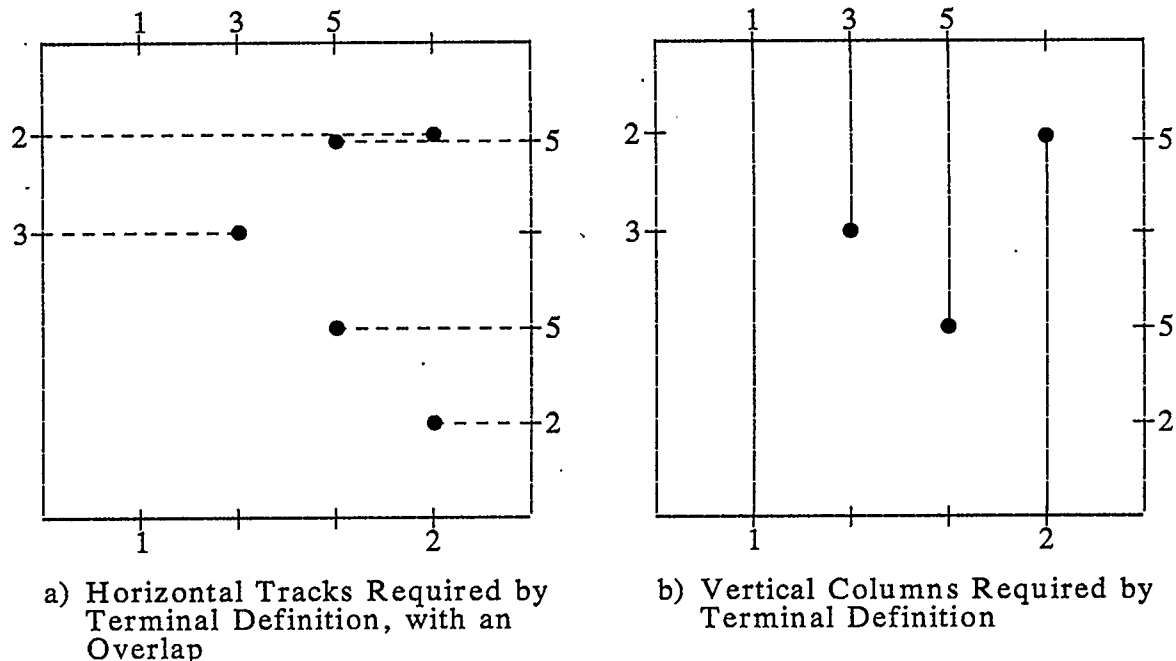


Figure 2.12 Terminal Constraint on a Switchbox Problem

The loop area routing scheme [Mukh86] routes an expanding model of the switchbox problem, proceeding from the centre of a routing channel towards the boundaries adding tracks and columns as they are required. Figure 2.13 depicts the solution to an example problem. This method does not use the common two-direction-two-layer routing model and routes by selecting the shortest nets to route near the centre of the routing region. This scheme cannot route more than one assignment per track and because it adds an equal number of tracks and columns as it proceeds, it produces square results regardless of the true dimensions of the routing area.

Another model that differs from the switchbox model is the three-sided channel [Souk81]. A three-sided channel has fixed terminals on three sides of the channel and floating terminals on the fourth side.

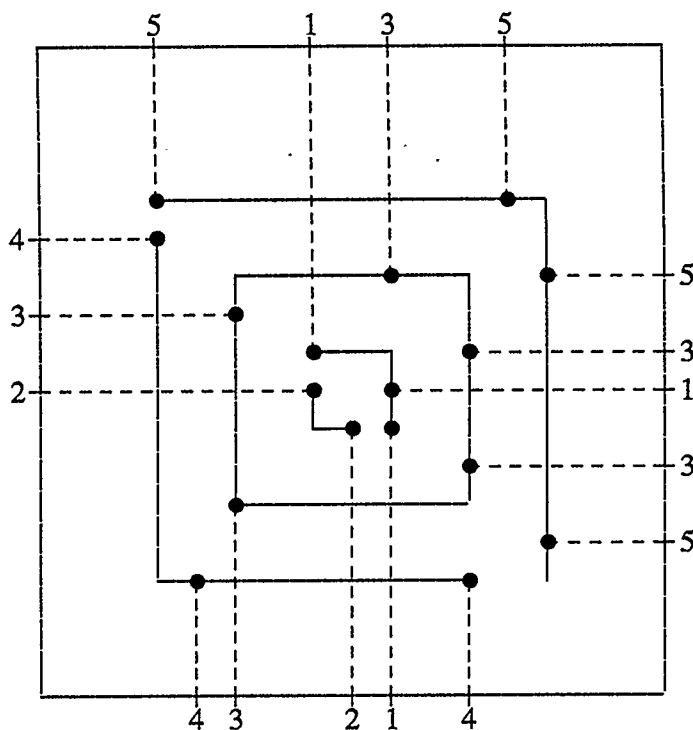


Figure 2.13 Loop Area Routing Scheme Example



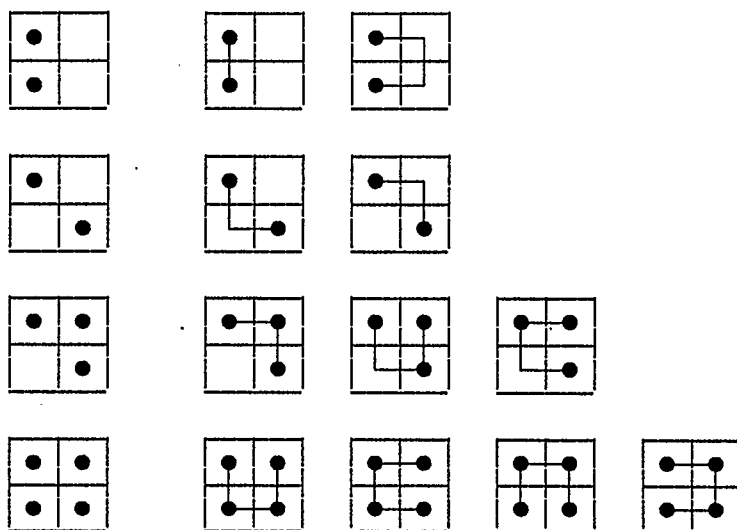
Three-sided channel routing methods route from the fixed terminal side towards the floating terminal side. Nets entering the channel on the third side automatically get the track that their terminal is on. Exiting nets are assigned a fixed position once routing has finished and are passed as constraints to the adjacent channel. Thus three-sided routing is not a truly independent routing model, since the terminal assignments are passed from channel to channel and terminal assignments may affect adjacent channels adversely.

The hierarchical pattern router developed by Burstein [Burs83], is capable of routing the switchbox model. The method subdivides a larger switchbox problem into a  $2 \times N$  grid model. The problem is then further subdivided into simple  $2 \times 2$  routing problems to which simple patterns can be matched (see figure 2.14a). These patterns are individually applied to advancing steps of the solution towards the final goal of solving the  $2 \times N$  problem. The steps of an example problem are shown in Figure 2.14b. The router can almost complete a difficult switchbox problem known as Burstein's Difficult Switchbox Problem (see Figure 2.15). Although net 24 is unrouted, the routers partial success gives it high marks for its switchbox routing capability.<sup>2</sup> This method routes entire nets one at a time and this has previously been shown to hinder 100% routing completion. The router uses no other heuristics other than these specific patterns from which to choose routes.

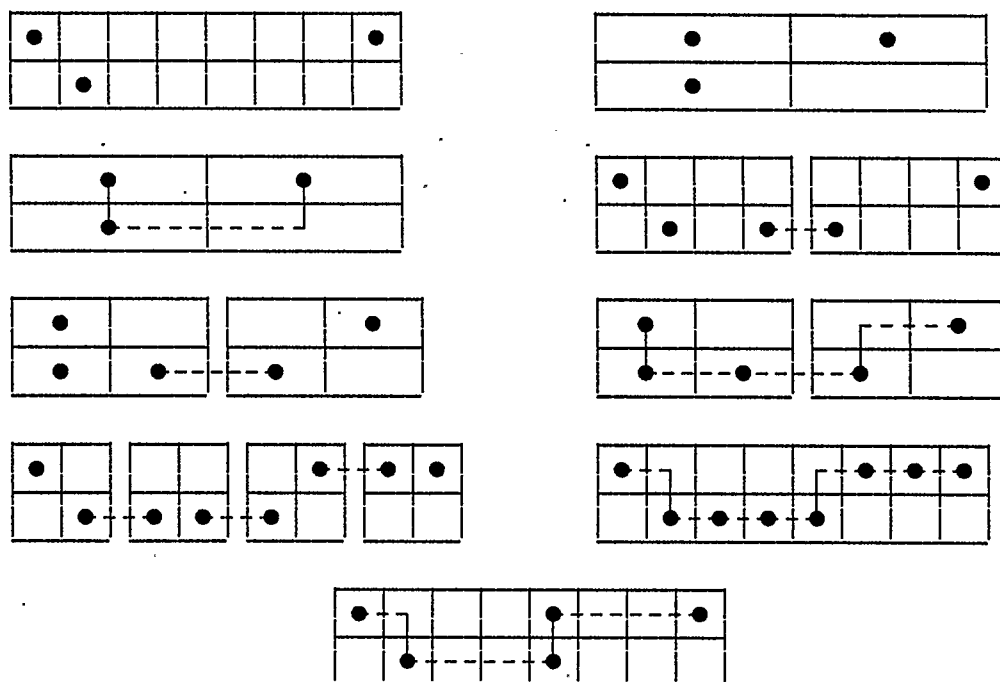
The column sweep channel approach can also be used to route switchbox problems, because the number of columns and tracks is known beforehand [Hama84] The router sweeps across the switchbox in the same

---

<sup>2</sup>Originally perceived as unsolvable, the extreme difficulty of Burstein's Difficult Switchbox thwarted many attempts at finding a solution until an automatic router produced one. The success of Burstein's pattern router has also been refuted by Deas [Deas86] as it orders the nets to be selected before starting the routing process to ensure the routability of a switchbox.



a) Basic Patterns for 2x2 Grid Routing



b) Solution to a 2xN Wiring Problem

Figure 2.14 Pattern Routing Example

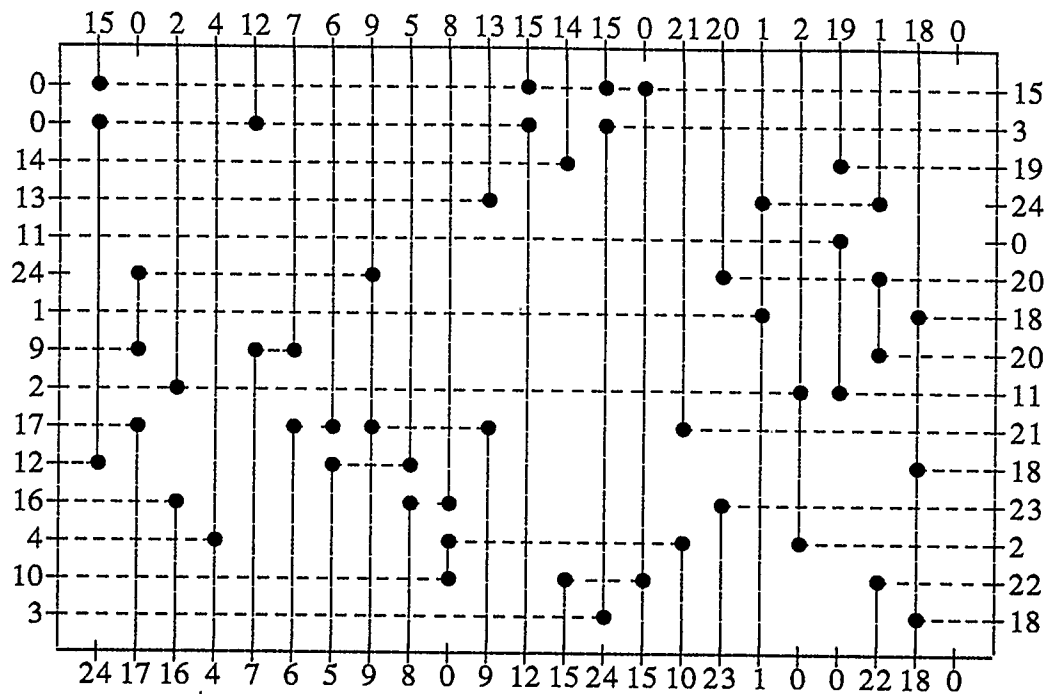


Figure 2.15 Burstein's Solution to his Difficult Switchbox Problem

manner as a three-sided channel router, except that the fourth side has fixed terminals and therefore, the router constrains the exiting routes to the tracks that they require. It is also one of the first methods to allow pre-routed nets. The 'greedy' switchbox router as it is called has also been benchmarked against Burstein's Difficult Switchbox Problem (Figure 2.16); however, this method suffers from the same drawbacks as the channel routing version does. It gives priority to nets on the basis of their location on the boundary of the switchbox. Nets routed earlier can block later nets which is shown by the fact that net 2 is pre-routed in Figure 2.16.

Marek-Sadowska's switchbox router is a heuristic algorithm that routes the most constrained nets in the switchbox in an effort to complete 100 per cent of all routes. The expansion of nets from the terminals is classified as either convergent, semi-convergent, or divergent (see Figure

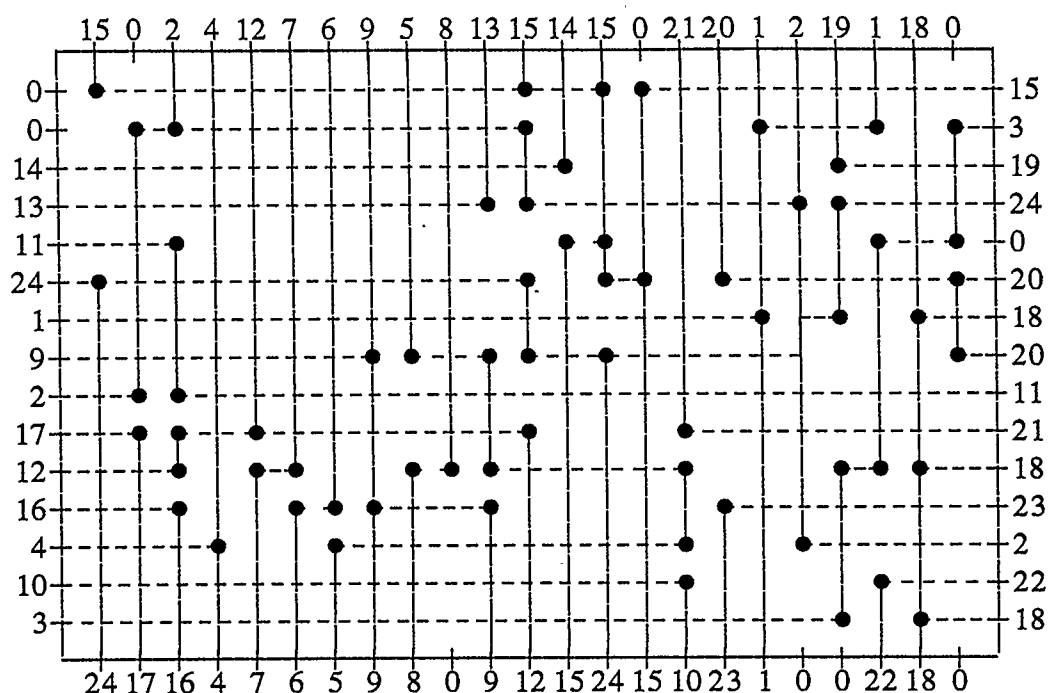
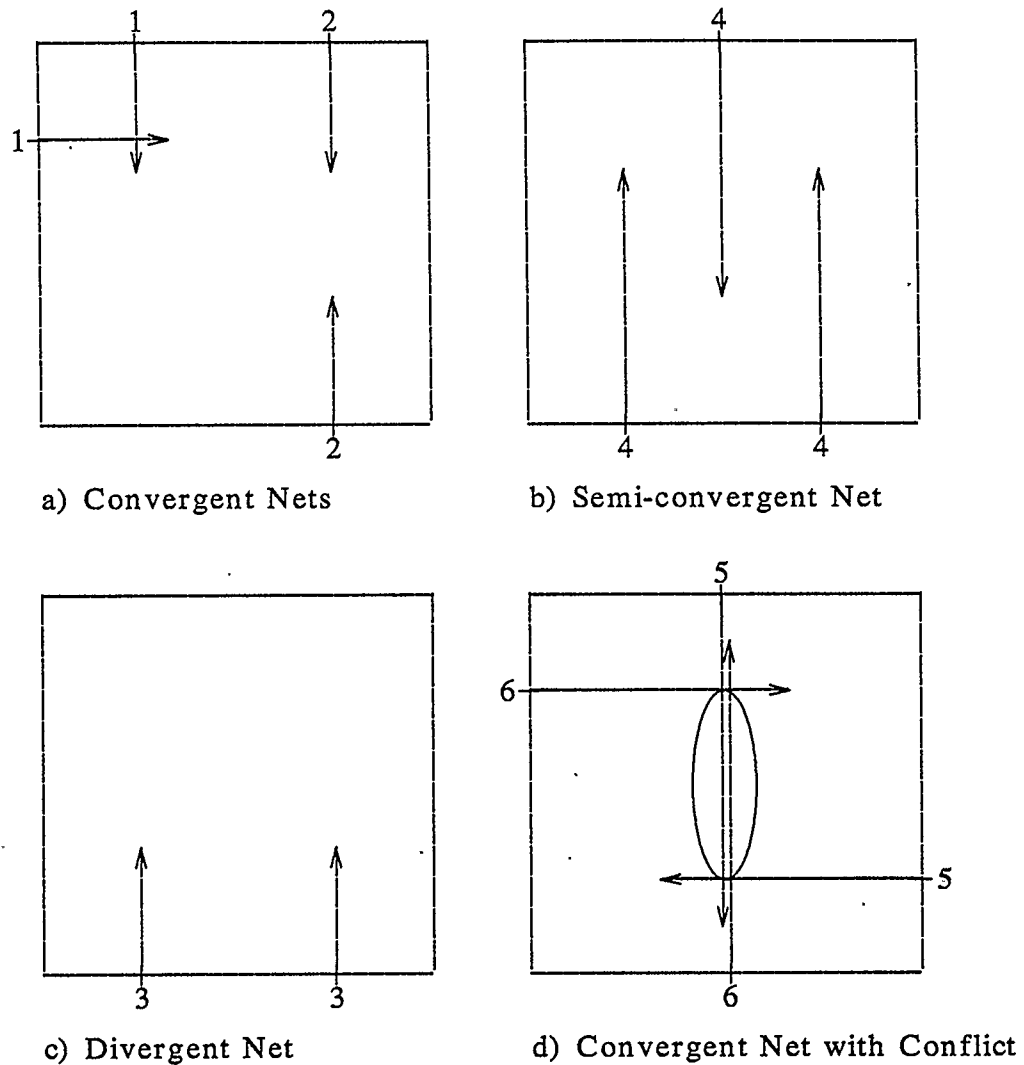


Figure 2.16 Hamachi's Solution to Burstein's Problem

2.17). Convergent nets have expansions that intersect either straight on or perpendicularly. Semi-convergent expansions travel in opposite directions, but do not intersect. Divergent expansions travel in the same direction and also do not intersect. She classifies nets as conflicting nets if their expansions overlap. For example, in Figure 2.17c net 5 conflicts with net 6. Her heuristics route nets based on these classifications.

Nets without conflict are routed first, and among these non-conflict nets convergent nets are the highest priority because the location of the routes are more constrained than the semi-convergent and divergent nets. Straight convergent routes are constrained to take a track by the presence of two terminals at either end of that track. Perpendicular and convergent routes are constrained to take a vertical and horizontal track by the presence of a terminal at the end of each track. In contrast, semi-



An arrow in this figure as in the figures that follow represent proposed paths for the route of a net. The representations of routed nets, contact cuts, and routes in different layers still stand as defined in Figure 2.6.

Figure 2.17 Marek-Sadowska's Net Classifications

convergent and divergent nets are similar to a channel route's horizontal segment; both have one segment that is not constrained by the presence of any terminal at either end of the segment, therefore it has a greater choice as to where to route.

Once a net is routed, or partially routed, Marek-Sadowska classifies the endpoints of routed expansions with respect to status in completing its route. A hanging terminal or pin is the endpoint of a semi-convergent or divergent route and can expand in the current direction away from the routed endpoint. A corner pin is the endpoint of a perpendicular and convergent route (a corner) and can expand both horizontally and vertically away from the endpoint.

After a net is routed, constraint propagation is performed. Each hanging pin is checked to see if it is blocked by a routed net. A hanging pin is blocked if it intersects perpendicularly with a routed net (see net 2 in Figure 2.18) or the tracks on either side of the hanging pin have been routed (see net 9 in Figure 2.18). In both cases, the hanging pin is propagated to an available track where it can complete its route.

Nets with conflicts are routed after the non-conflict routes are completed. Using the same priorities, convergent routes are routed first, and semi-convergent are routed second again owing to their greater flexibility in the choice of routes. Divergent nets are routed last, although

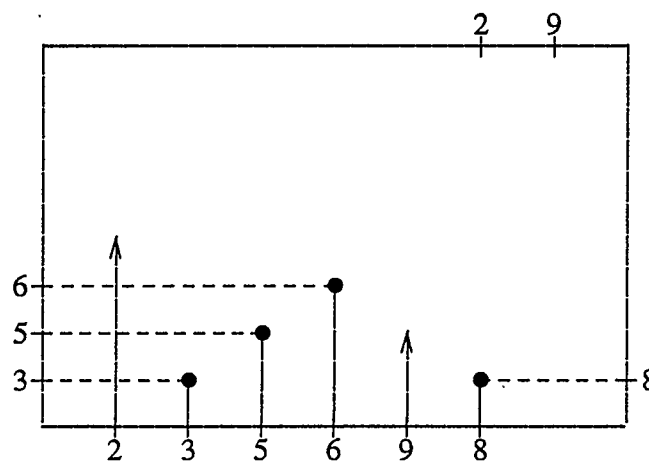


Figure 2.18 Constraint Propagation

their net length increases the farther they expand away from the border of the switchbox. Once one conflict net is routed, the other hanging pins are propagated to available tracks. For example, in Figure 2.19 net 1 is routed arbitrarily before net 2. The left terminal of net 2 moves around net 1 by turning down on an available track and its lower terminal will turn left on an available track.

Her heuristics do decide how to resolve conflicts between nets of different classifications, but they decide arbitrarily between routes of the same type, for instance between two straight connections. Her heuristics have also been benchmarked against Burstein's Difficult Problem (Figure 2.20).

Joobbani's WEAVER routes switchboxes using heuristics similar to Marek-Sadowska's for constraint propagation (pattern routing) and for routing non-conflict convergent routes (corner filling). However, it uses heuristics borrowed from channel routing to resolve conflicts between nets.

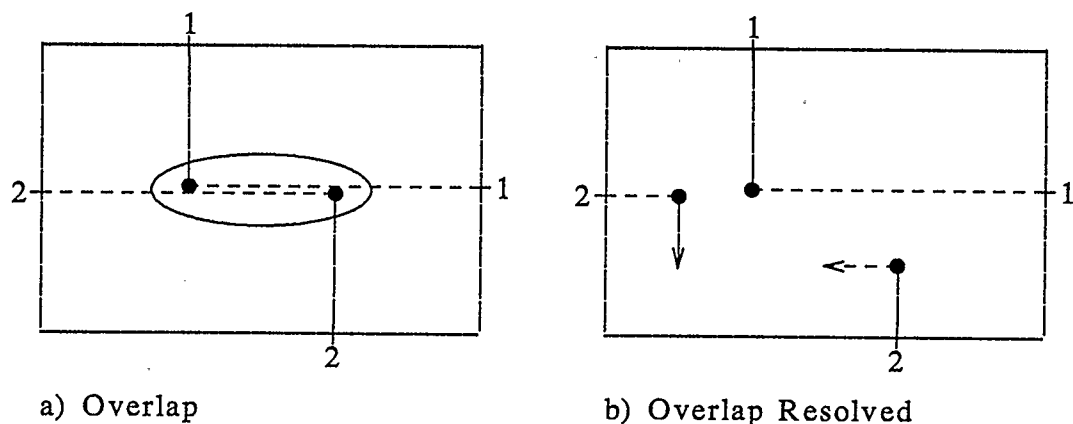


Figure 2.19 Expansion Direction Changed by Constraint Propagation

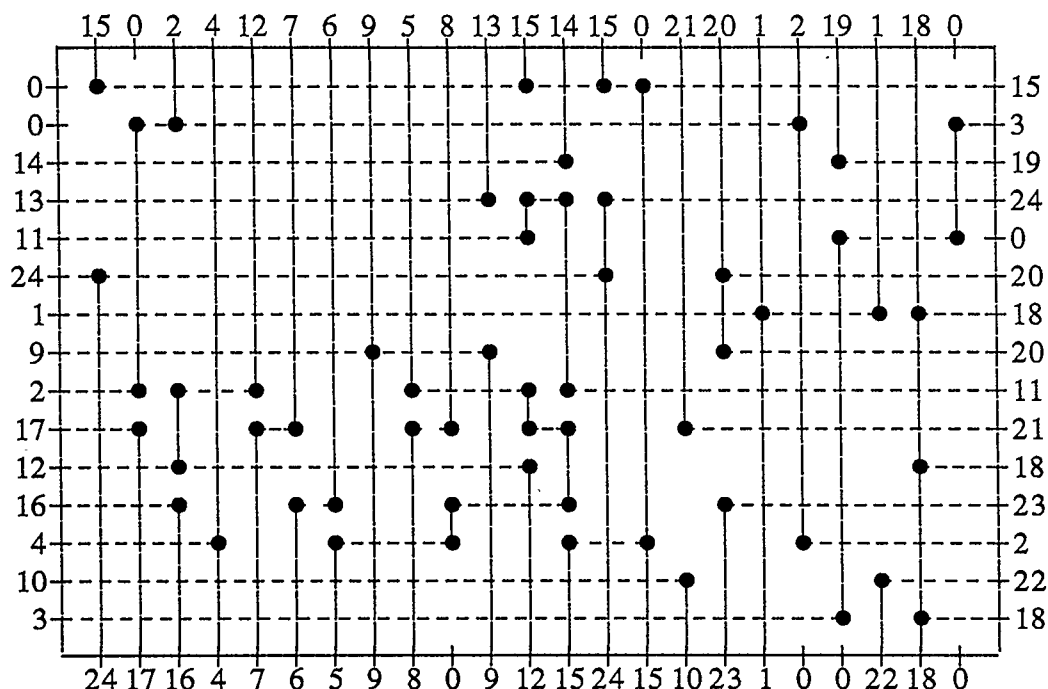


Figure 2.20 Marek-Sadowska's Solution to Burstein's Problem

Joobbani's procedure first generates a Steiner tree as an estimate of the net length and the path that a net may take through the switchbox. In Figure 2.21, a minimal rectilinear spanning tree and a minimal rectilinear Steiner tree have been generated for the same set of points. A minimum spanning tree is generated by iteratively connecting a point to the growing tree if the link required to connect the point to the tree is the link of minimum length at that iteration. Steiner trees [Hana66] are different from minimum spanning trees by the presence of extra points called Steiner points that further decrease the length of wire used to interconnect the terminals of a net (see Figure 2.21).

The estimate of net length is improved by applying vertical and horizontal constraints to the nets, minimizing channel density, and maximizing net merging. Similar to Eustace's least cost path algorithm,



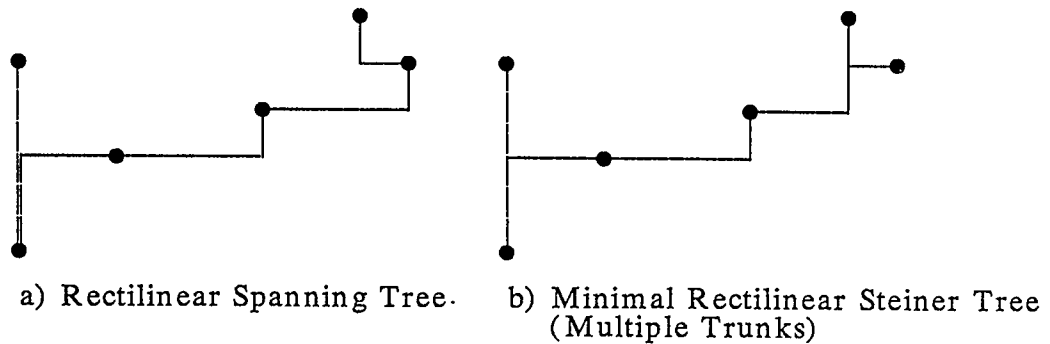


Figure 2.21 Minimum Rectilinear Spanning and Steiner Trees

leaf nodes in the vertical constraint graph are selected to route on an upper track in the switchbox. Next, the nets chosen are merged in different combinations following horizontal constraints. The congestion heuristics select the nets that travel through the most congested part of the switchbox. The same is done for the lower leaves in the vertical constraint graph. Once a net is fully interconnected, heuristics remove any unnecessary vias in the route. Heuristics that find equivalent minimal rectilinear Steiner trees are applied to nets whose proposed paths now overlap the routed nets. A set of control heuristics govern the sequence in which the Steiner tree, congestion, via, merging, and vertical and horizontal constraint heuristics are applied to a routing problem. The WEAVER can route both channels and switchboxes.

The WEAVER has been tested against Burstein's Difficult Switchbox Routing Problem (see Figure 2.22) and other channel routing problems to show its expertise at routing. The system is able to route channels better than the theoretical minimum, because wires are permitted to overlap on corners and in parallel (see Figure 2.23) which is a radical departure from the two-layer-two-direction wiring model. The WEAVER is implemented as an expert system and uses an unprecedented number of heuristics, over

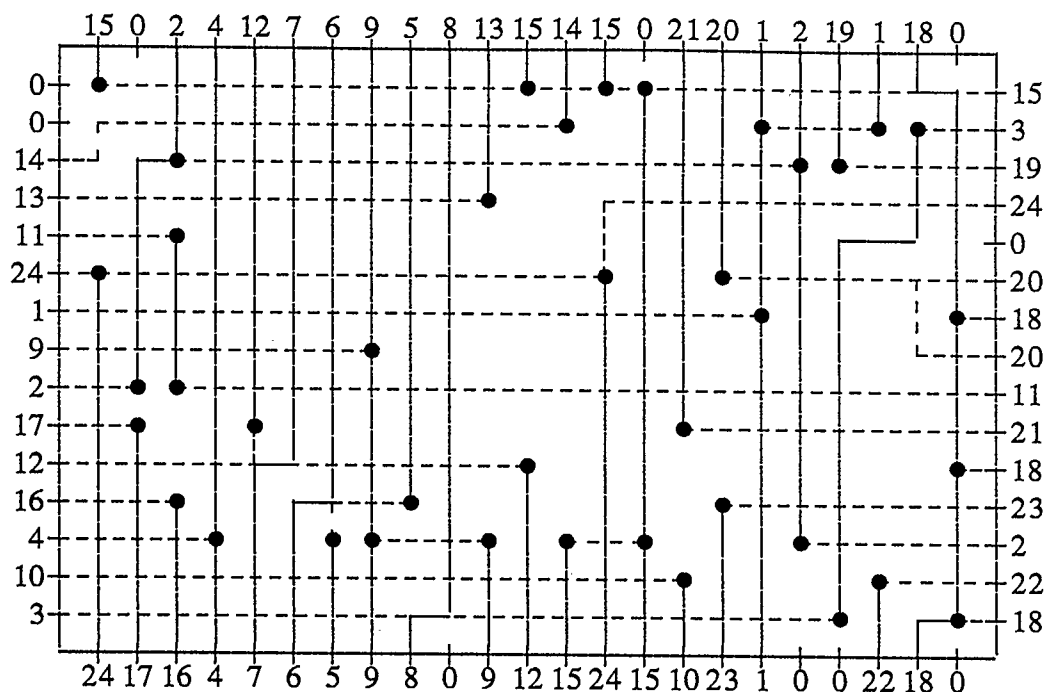


Figure 2.22 Joobbani's Solution to Burstein's Problem

700, most of which have been taken from human routing experts, to solve the routing problem.

To sum up, switchboxes are similar to channels in that they both contain no obstacles in their interior and they use the grid approach and two-layer-two-direction wiring model; however, they differ from channels in their definition and the methods used to solving them. Terminals can be placed on all four sides of the switchbox, which restricts the routing region from being expanded during routing. The loop area routing scheme expands the routing region, but without regard to its actual physical dimensions. Two switchbox routers route net by net (pattern routing) and column by column (column sweep approach). The two most successful switchbox routers, Marek-Sadowska's and Joobbani's, route from the boundaries inwards interconnecting the most constrained nets

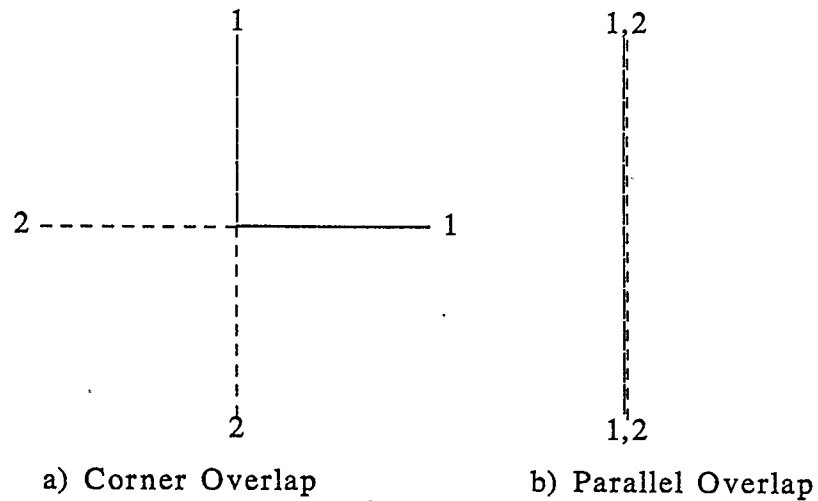


Figure 2.23 Corner and Parallel Overlap

first which is in keeping with the goal of attaining 100 per cent routing completion. The most constrained nets in both system are corner and straight routes and nets that are blocked by other nets and are to be propagated to an available track. Each method proceeds differently to route nets which conflict with other nets. Marek-Sadowska orders the conflict routes based on the same classifications that were established for non-conflict routes. On the other hand, Joobbani uses heuristics common to channel routers to interconnect conflict routes. Although both routers use different techniques, they both expand nets to find out where proposed routes lay with respect to one another, and both alter the proposed paths once they overlap with routed nets. Joobbani's system gets closer to achieving 100 per cent routing completion by allowing nets to overlap in parallel and on corners. Marek-Sadowska's algorithm relies heavily on constraint propagation to achieve this.

## 2.6. Other Routing Techniques

The routers discussed above are heuristic algorithms. They apply a heuristic to a partial routing solution when the conditions of the heuristic match those present in the problem state and advance the solution one step further towards completion. Alternative routing techniques have been developed to expose more than one possible routing solution and evaluate the quality of the solutions, and choose the one having the best quality. Two of these techniques discussed below are the branch and bound technique and simulated annealing.

The branch and bound routing technique [Kern73] keeps track of different partial routing solutions as each proceed towards a different final solution. A bound is established as to the number of iterations that the search can perform, and when the bound is reached, the quality of each partial routing solution is evaluated. The partial solution with the highest quality is assumed to be the best path, and the search for the final solution continues from that path onwards. The other partial solutions are discarded. A bound of infinity leads to a total search; a bound of 0 leads to a depth first search--taking the first partial solution that comes along. Trial and error establishes what bound is required to obtain a good solution.

Another routing technique uses simulated annealing [Vecc83] [Kirk83] [Sech85], which is a randomized heuristic algorithm.<sup>3</sup> A randomly generated final solution to a routing problem is permuted into new variations that are compared in their quality to the previous one. A temperature parameter provides an upper bound to the increase in cost

---

<sup>3</sup>Simulated annealing is based on a technique used in physics and chemistry to restore crystal structure in semi-conductor materials and produce stable chemical compounds. The material is heated to extreme temperatures forcing molecules into rapid random motion. The compound is then cooled slowly allowing molecules to settle in regular patterns that are highly stable.

that a new solution can have over the previous one. If the cost of the new solution is lower, it is always taken. However, if it is higher, its change in cost is compared against a randomly generated value. If it is lower than this value, the new solution is accepted. The random variable controls the amount of hill climbing moves (moves which are greater in cost) that are introduced to system. When the temperature is high, almost all hill climbing moves are accepted; as the temperature is lowered, less and less are. Theoretically, if the temperature decreases at a satisfactorily slow rate at the start, enough permutations of the solutions are generated to guarantee that an optimal solution is found. Trial and error is used to establish what that rate is.

Heuristic algorithms have several advantages over the other two search strategies discussed above. First, a heuristic algorithm does not require extra memory and state saving techniques for operation as does the branch and bound technique. Because NP complete problems such as routing must explore a large area of the problem space before they know if one path is better than another [Gare79] [Baas78], the amount of resources required would be extensive. Second, heuristic algorithms can use a multitude of heuristics to choose how to proceed from one partial routing solution to the next. In contrast, simulated annealing uses one randomized heuristic. Many permutations of final solutions have to be generated before the technique converges on a good solution, a process which can take up to several hours for a fairly difficult routing problem.

It would seem that because heuristic algorithms do not probe a large portion of the search space, they are more in danger of pruning a path containing a good solution than the other two methods. However, the use of specialized heuristics protect a partial solution from being lead to a dead end for most routing problems.

## 2.7. Discussion of Current Routing Methods

Among the routers presented in chapter 2, the discussion of current routing methods focuses on switchbox routers. Most of the issues concerning channel routing have been investigated and there are highly competent routers available to solve the channel routing problem. Switchbox routers are not as well-researched as channel routers and there is a great need in CAD systems for good switchbox routers. The discussion of switchbox routers concentrates on Marek-Sadowska's and Joobbani's systems, because they achieved the best results on the difficult switchbox routing problem. Three general problems are mentioned.

First, from the discussion of Marek-Sadowska's switchbox router, it was noted that her router arbitrarily chooses to route conflict nets based on their classifications. Convergent nets are routed first, then semi-divergent nets, then divergent nets. Although classifications do signify how constrained each class of nets is to take a route, they give no indication of how nets are able to expand in all directions. This is shown in the inability of her heuristics to discover if one net can move around another when they conflict with each other. In Figure 2.24a, the net expansion for a switchbox problem has been performed. The leftmost terminal of net 1 does not route on a corner with the upper terminal, because of the conflict with net 2; however, it extends across in a corner route with the right most terminal of net 1, because neither of those endpoints conflict head on with other nets. The result of that choice is shown in Figure 2.24b; net 2 remains incomplete and neither of its terminals can expand towards an available track. The basic problem is that Marek-Sadowska's heuristics expand terminals and perform constraint propagation in one direction for each net. An expansion is only made in line with the direction of the routed segment for each net. Constraint

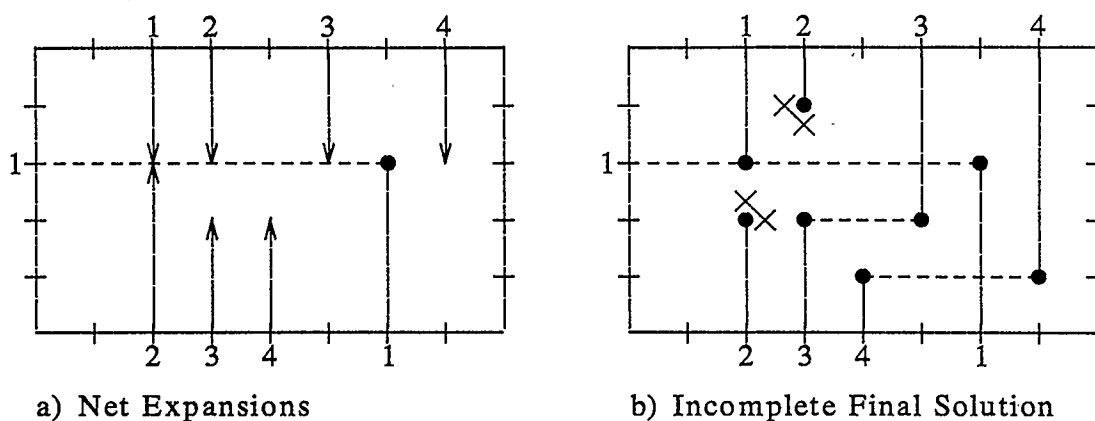


Figure 2.24 Overlap Dilemma

propagation only checks the available tracks in front of the net as well. Semi-convergent routes and divergent routes must turn their routes at some point to connect with the rest of the net; but performing net expansion and constraint propagation in one direction does not provide information on how well the nets can route in the opposite direction when the time comes. This thesis proposes a new set of heuristics that expands terminals and performs constraint propagation in two directions rather than one and decides which nets may route based on how the expansions conflict with other nets.

Second, between the two systems, no general heuristics are provided to lay down routes other than by using the two-direction-two-layer wiring model which has been shown to hamper switchbox routers in obtaining 100 per cent completion. Joobbani provided a partial solution by using heuristics to minimize vias after nets were routed and allowing parallel overlaps between nets if it is necessary to complete a solution. However, no general group of heuristics have been provided to establish when and if parallel overlap should be used on a routing problem. This thesis builds upon the heuristics proposed above by suggesting general heuristics that

can be used when the expansion heuristics fail to find routes for nets using the two-direction-two-layer wiring model.

Finally, the successful automated routers, particularly the switchbox routers use many and varied constraints and apply them as heuristics to the routing problem. However, there is still no cohesion between the heuristics that are used to route channels and those used to route switchboxes. Joobbani's router solves both routing problems, but uses different heuristics for each. Within the switchbox routing problem, Joobbani uses one set of heuristics, constraint propagation and corner filling, to route non-conflict nets and uses another set derived from channel routing techniques to route conflicting nets. In contrast, Marek-Sadowska uses the same net classifications and priorities to route nets with or without conflicts. Although Marek-Sadowska's router does not route channels, there is evidence to suggest that her heuristics may be able to. Her heuristics are as capable as Joobbani's at routing conflicting nets and Joobbani's conflict heuristics were based on channel routing heuristics. *It is proposed in this thesis that the net classifications and heuristics created by Marek-Sadowska and the other proposed heuristics can be used to route both the channel and switchbox routing problems, thereby uniting both models through using this general routing method to solve them.*

Chapter 3 discusses these three points in detail and presents the method and the heuristics that are used to solve them.



## **CHAPTER 3**

### **New Heuristics for VLSI Routing**

This chapter concentrates on the new heuristics that have been developed as a solution to the problems outlined at the end of chapter 2. Section 3.1 discusses the background work that led to the development of the new heuristics. In section 3.2, the new heuristics that define available tracks, select nets to be routed, and propagate constraints are outlined. Section 3.3 details the heuristics that route corner overlaps in cases where nets cannot be completely interconnected using the above heuristics. The final section explains how these heuristics can be used to solve the channel routing problem.

#### **3.1. Background Work**

The heuristics that are described in the following sections are based on Marek-Sadowska's methods for solving the switchbox routing problem. This section explains how this decision came about.

It has been mentioned previously that a switchbox router's main goal is to route all nets to 100 per cent completion. Because Joobbani's system achieved this goal in nearly all his examples, his general channel routing heuristics were used as a base on which to develop heuristics to solve the problem outlined in Figure 2.24. The main thrust of the work was to develop heuristics that ensured that those routes which passed through the most heavily congested switchbox areas were given the highest priority to route [Keef86]. Heavily congested areas were defined to be where the areas of maximum horizontal and vertical density overlapped and the nets that passed through the area were routed first. The nets outside of the area

were ignored to the detriment of finishing their own routes. Heuristics were added to include their expansions in the routing of the congested area, and it became evident that this method was performing the same techniques as Marek-Sadowska's heuristics for switchbox routing, but in a convoluted way. Corner filling was performed automatically, as in most switchbox problems the corners are the most congested areas. The key to solving the switchbox problem lay in recognizing when nets were allowed to expand to proposed routes and noting how and where conflicts occurred. Once nets were selected to route, it was also important that blocked nets propagate to the routing space still available and new expansions made. This procedure had already been established by Marek-Sadowska.

The tactic was changed to use Marek-Sadowska's switchbox heuristics as the base on which to expand and find heuristics that could solve the problem in Figure 2.24 using the same approach of net expansion, conflict resolution, and constraint propagation. Particular attention was paid to the way her heuristics chose nets to route based on the shape of their paths of expansion. From here, it was hypothesized that not enough information was being gathered from the problem state by allowing nets to expand in only one direction (except for corner routes) and by checking blocked routes ahead of the one path. New heuristics were created to allow nets to expand in two directions towards their destinations and to check for blocked paths in both directions. Heuristics were also created to solve a situation when a net could not expand in both its desired directions. These heuristics are presented in the following sections.

### **3.2. New Heuristics for Routing Switchboxes**

The heuristics outlined are based on definitions created by Marek-Sadowska for net conflicts, terminal types, net expansions, and constraint

propagation. Each is discussed with respect to how they have been changed to support the new method for routing switchboxes.

The first change affects the creation of hanging pins for conflicting nets. Previously, Marek-Sadowska allowed perpendicular routes to interconnect if they did not have a conflict with other nets. The new definition requires that only perpendicular routes which intersect with the closest terminal to their location can route if they have no conflicts. Closest is defined to be either the least distance between the x coordinates of two terminals expanding in a horizontal direction or the least distance between the y coordinates of two terminals expanding in the vertical direction. This changes how net 1 is routed in Figure 2.24. The upper terminal of net 1 and its leftmost terminal form a perpendicular route, but the upper terminal of net 1 has a conflict with net 2 and cannot be connected. In Marek-Sadowska's version, the leftmost terminal would continue on to connect to the far right terminal because both their paths intersect and neither has a conflict. By the new definitions, the left most terminal would not be allowed to continue on past its closest terminal (the upper terminal) although it has no conflict. The result of this definition, is that net 1 will not automatically take a horizontal track that net 2 may require, a need that was established by its conflict with the upper terminal of net 1. It follows from this new definition that nets perpendicular to a net with a conflict must also be considered to have a conflict, since the routing of one net helps establish the routing for the other. The definition for hanging terminals has been widened to include those nets which have a conflict as defined above. This is in keeping with Marek-Sadowska's general definition of hanging terminals, which defined hanging pins to be the endpoints of nets with conflicts. Usually these are the semi-convergent and divergent routes, and nets with head-on conflicts, but now nets whose

expansions are perpendicular to a conflict are included.

The definition of net expansions was also altered to define which directions and to which terminals hanging pins were now allowed to expand. For nets with two terminals, each terminal was allowed to expand horizontally and vertically towards each other. For multi-terminal nets, each terminal at the ends of the switchbox expanded towards its closest neighboring terminal in either direction, and each terminal in the centre expanded towards the closest terminals on either side of its position. Allowing a terminal to expand towards its closest neighbors only gives a good estimate of the minimum net length required to interconnect the terminals in a similar fashion to the way minimum spanning trees are created to interconnect terminals.

The definitions governing net expansions were also changed to reflect how close the expansion from a hanging pin could get to its target by discovering which tracks were available in between. An available track for the expansion of a route is defined as a track that is perpendicular to the expansion and does not have a routed segment intersecting the expansion or routed segments on that track extending in either direction away from the expansion. Using the downward expansion of net 4 in Figure 3.1 as an example, three tracks are shown as being available to net 4. The first three tracks are not available because of the presence of routed segments that are perpendicular to it. The next two tracks are available and are labeled A and B. The sixth track is not, because net 4 must turn right at some point to connect to its other terminal and there is a routed segment proceeding to the right away from it. The next track, labeled C, is available.

In any expansion, there can be more than one available track that a terminal can expand towards. An additional constraint in the definition

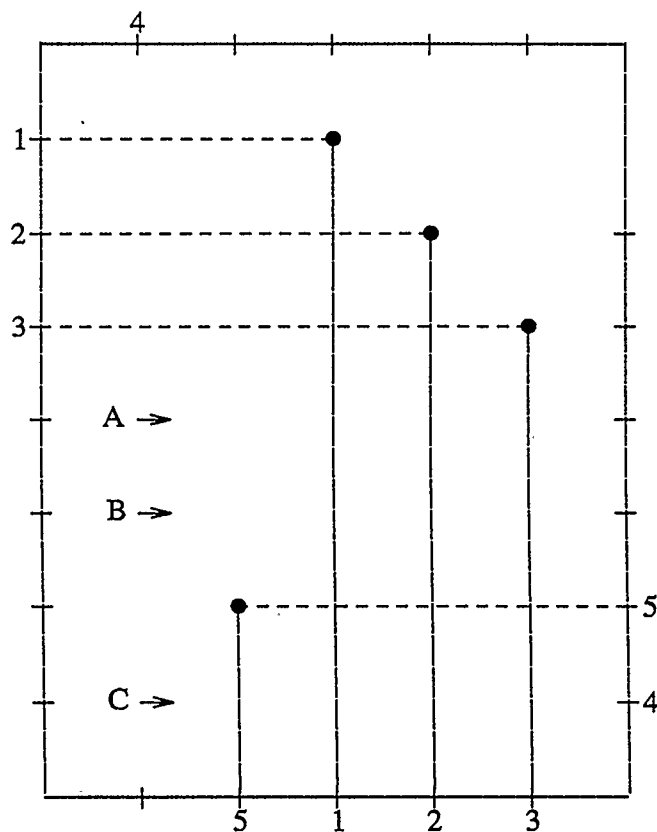


Figure 3.1 Available Tracks

pushes a hanging pin's expansion on to the farthest track available. An example route is shown for net 1 in Figure 3.2 in which it takes the first available track at each expansion (Figure 3.2a) and takes the last available track (Figure 3.2b). Choosing the first available track creates a staircase route and is undesirable because corners in a route produce high electric fields. At 1 micron technology, an effect called metal migration is produced due to the small cross sectional area of the route and may compromise the route's integrity. To circumvent this 'greedy' approach to selecting tracks, the farthest track available is selected. This ensures that the expansion extends as far as possible in the direction of its destination knowing that closer alternatives may be available if needed.

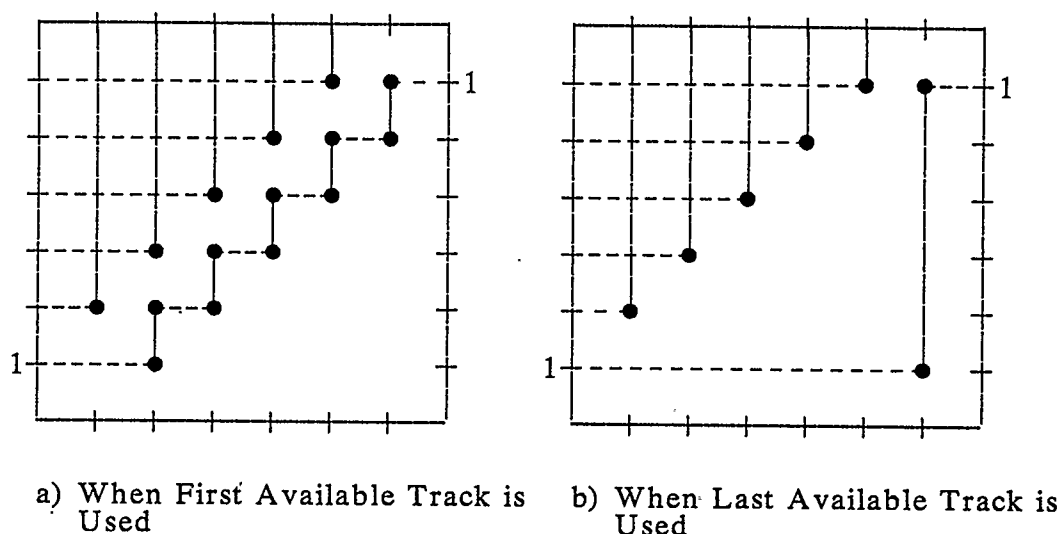


Figure 3.2 Choosing the First Versus the Last Available Track

Similar to the way constraint propagation is performed by Marek-Sadowska, nets which cannot find an available track in one direction are propagated to an available track in the other direction. With the new net expansion definitions, nets are now automatically checked to see if they can expand in both of their proposed directions.

The following heuristics resolve net conflicts based on what net expansions are produced for all hanging pins. Following Marek-Sadowska's definitions closely, a conflict is defined to be a connection (or route) that intersects with another in *parallel*. This includes overlaps with the endpoints of routes. With one exception, routes that intersect and are perpendicular to each other are not conflicts. If an expansion, expansion 'a', intersects with another expansion, expansion 'b', such that they are perpendicular to each other and the end point of path 'a' intersects with expansion 'b', then the 'a' is said to have a conflict with 'b' (see Figure 3.3). Because expansion 'a' finishes with its path on 'b', 'a' may at some

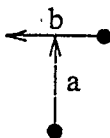
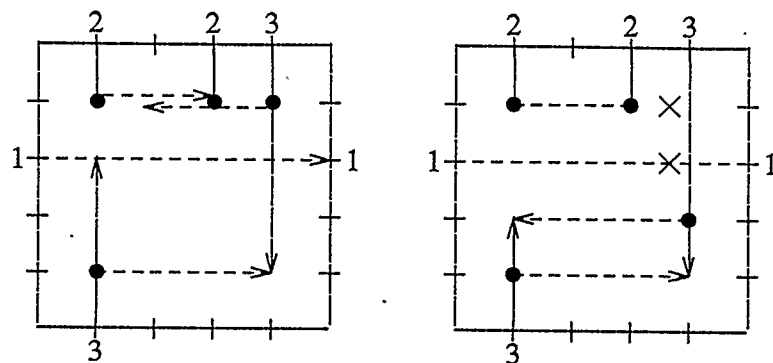


Figure 3.3 Example of Net Conflict

future time want to route along the track that expansion 'b' is on presently. The conflict between the two expansions means that, for 'a' at least, this track may not be available.

Once the conflicts have been identified, there remains the arduous task of deciding which expansions should route and which should not. Three main heuristics based on Marek-Sadowska's net classifications and heuristics for net selection are outlined below. These heuristics are presented in the order of priority that they are applied to solve the routing problem.

- (1) If two hanging terminals are connected straight through and have no other straight through expansions overlapping with them in a parallel direction, the connection is routed. All other types of routes that may overlap with the net are removed (see net 1 and 2 in Figure 3.4).
- (2) If a hanging terminal has two expansions and one has a conflict, but the other does not, then the non-conflicting expansion is routed (see upper terminal of net 3 in Figure 3.4a). The other expansion is removed.
- (3) If a hanging pin has two expansions neither with a conflict, the one which expands in the same direction as its routed segment is chosen. (see the lower terminal of net 3). The other expansion is removed from consideration for future routing.



a) Expansions of All Nets    b) After Nets 1 and 2 are Routed

Figure 3.4 Net Conflict Resolution Examples

Each time an expansion is routed and other overlapping routes are removed, constraint propagation is performed. All routes that may now overlap with the newly routed segment will be propagated in the same direction as the constraint propagation dictates. Once all terminals have been propagated to their new locations, each terminal expands again. Conflicts between routes are investigated, and again the three heuristics above are used to decide who wins and who loses a conflict.

The procedure outlined above constitutes one cycle in the routing process. This cycle of expansion, conflict detection, route selection, route deletion, and constraint propagation fits comfortably in the cycle used to route non-conflicting nets (net expansion, route selection, constraint propagation). Using this cycle to demonstrate how the above example would be routed, net 2 and 1 would be routed by the first heuristic and the horizontal expansion from the upper terminal of net 3 would be removed from consideration. Instantly, net 3's upper terminal would be propagated down past net 1, because its horizontal route is now blocked by net 2. Net 3 then expands horizontally to connect with the other expansion of net 3. Because both terminals of net 3 now have two expansions without conflicts



(from other nets), one of their expansions is chosen to route. In this case, the upper terminal of net 3 routes vertically in line with its routed segment and its horizontal expansion is removed. Now the lower terminal of net 3 connects with a straight connection vertically to the other hanging pin and its net is routed.

The heuristics are also demonstrated to show how they would solve the problem in Figure 2.24. Figure 3.5 shows the expansion of the four nets horizontally and vertically after each has been routed one unit into the switchbox.

Both terminals of net 2 have been propagated one unit farther because of the constraint created by the presence of net 1's and net 3's hanging pins. In keeping with the net expansion heuristics, the left most terminal of net 1 has expanded in three directions towards its other two terminals. Using the three net selection heuristics established above, the upper corner route of net 1 can be routed because it has no overlaps with other routes. When the other two perpendicular expansions are removed from consideration, net 2 can route its upper left corner. Once it has removed its other expansions from consideration, net 3 can take the second lowest

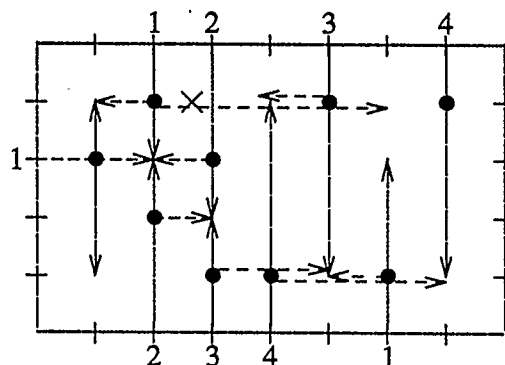


Figure 3.5 Net Conflict Resolution Using New Heuristics

track which makes net 4 free to take the lowest track, once net 1 expands upwards into the switchbox. Figure 3.6 shows the solution generated by the heuristics. Because the lower terminal of net 1 has moved up to the same height as its left most terminal, the upper terminal may now expand down and right to meet it.

A curious problem with these heuristics is that they can create hanging routes as is done in this solution. Net 1 has expanded down in a futile attempt to meet up with the lower terminal of net 1, which shows that even these heuristics are too short sighted to see where every expansion can lead.

### 3.3. Corner Overlap

The heuristics outlined above may be used to establish paths of expansion for routes, but they do not handle the case where routes cannot expand in their desired directions. This situation occurs when the two choices of a route's expansion are met head on by connections that are already routed on those tracks. For 100 per cent routing completion to be achieved, the new route will have to overlap one of the old routes. Heuristics to route corner overlaps have been developed to facilitate this.

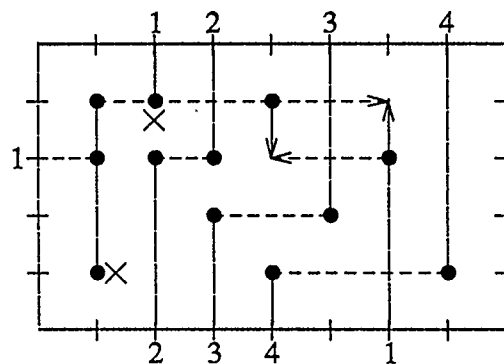
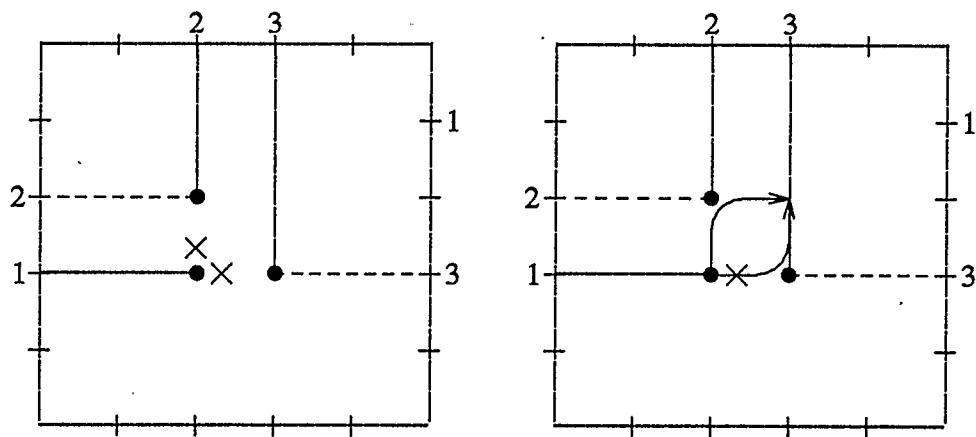


Figure 3.6 Final Solution

However, heuristics for parallel overlap are not developed in this thesis. Parallel overlap is usually avoided in routing because of the increased capacitance created between parallel wires.

When a hanging terminal cannot expand in either direction, it seeks to overlap with the corner of one of the routed connections. Figure 3.7a depicts a situation where net 1 cannot expand in either direction towards the other terminal. Figure 3.7b shows that net 1 can establish a corner route with net 2; however, routing on net 3 creates a parallel overlap. Because parallel overlap is undesirable, it would be necessary for net 1 to expand down, away from the conflict and attempt to find an escape route to an available track where it can complete its route.

The switchbox model for this problem uses the two-direction-two-layer wiring by default. When corner routes are introduced into the problem, the default layers from the two routes establishing the corner route are switched at the corner of their routes. If one of these routes



a) No Straight Expansion Possible b) Corner Overlap Possible in One Direction

Figure 3.7 Conflict Resulting in a Corner Overlap

should intersect with another route, 'a', then route 'a' would switch layers. If in turn route 'a' intersected with another route 'b', route 'b' would then switch layers. This process has been christened 'corner overlap fanout', because the change in layers originates at the corner overlap and fans out across all affected routes.

The heuristics which create the fanout work as follows. First, the via is removed at the corner where both nets overlap and both nets are given a directive to fanout in one direction to create or remove the necessary vias so that the two-layer-two-direction model is once again restored. Figure 3.8a shows a situation where net 6 must place a corner route over net 5. Both nets will remove the vias at their corners and create an overlap fanout going in the same direction as net 6 took approaching the corner. Net 5 will fanout going bottom-up, and net 6, following the expansion path of its corner will fanout to the right (see Figure 3.8b).

Figure 3.9a depicts what would happen should net 6 and net 5 encounter other intersecting routes. Fanout for net 5 and 6 will halt when it finds

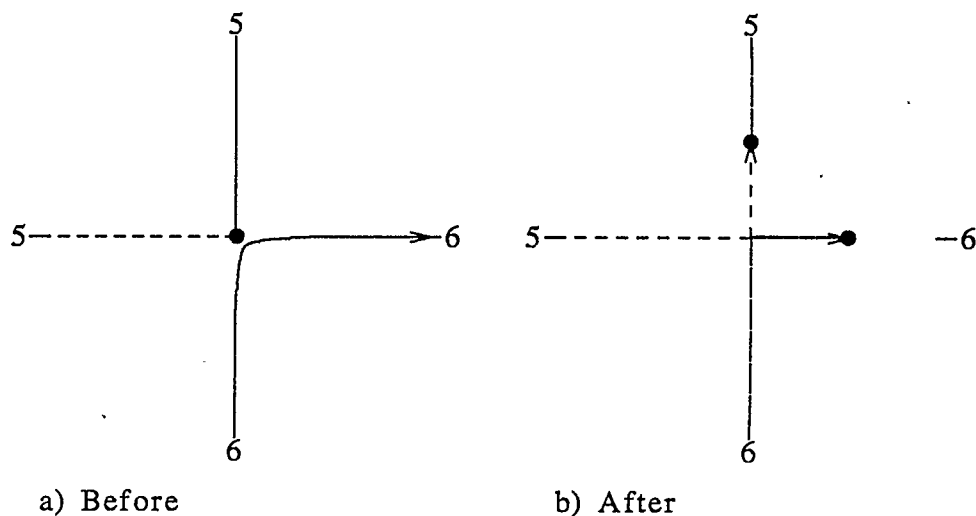


Figure 3.8 Simple Corner Overlap Example

the first free track on which it can place a via. In the mean time, every route that net 5 and 6 intersect with before that point will fanout as well. Arrows on the diagram indicate this action. Fanout arrows that meet each other going in opposite direction cancel their actions out. If net 5 and 6 should find that their routes bend on the available track, the via is removed from the corner and fanout is stopped. Figure 3.9c shows the final solution to the example. Vias are represented by bullets. The area that has been affected turns into an inversion of the layers originally assigned to each direction, but it is interesting to note that two-layer-two-direction wiring model is still intact.

### 3.4. Heuristics for Channel Routing

The heuristics discussed in the previous section deal exclusively with the switchbox routing problem, but this section discusses how the heuristics of terminal expansion, track availability, constraint propagation, and net conflict resolution can be used for the channel routing problem. Although this thesis has stated that current channel routers use different heuristics than for routing switchboxes, the similarity between both problems suggests that it may be possible to solve them both with the same heuristics. This section discusses that possibility in detail.

Although track availability and net conflict resolution have never been made an issue in channel routing problems, channel routing *has* used the four heuristics of net length, vertical and horizontal constraints, congestion, and merging to resolve conflicts between nets that could not be routed on the same track. The use of the Marek-Sadowska's heuristics along with the new heuristics may be useful for the channel routing problem as well. Consider the channel routing problem in Figure 3.10. Corner filling heuristics used in the switchbox routing problem are useless because there are no side terminals; however, the heuristics that perform

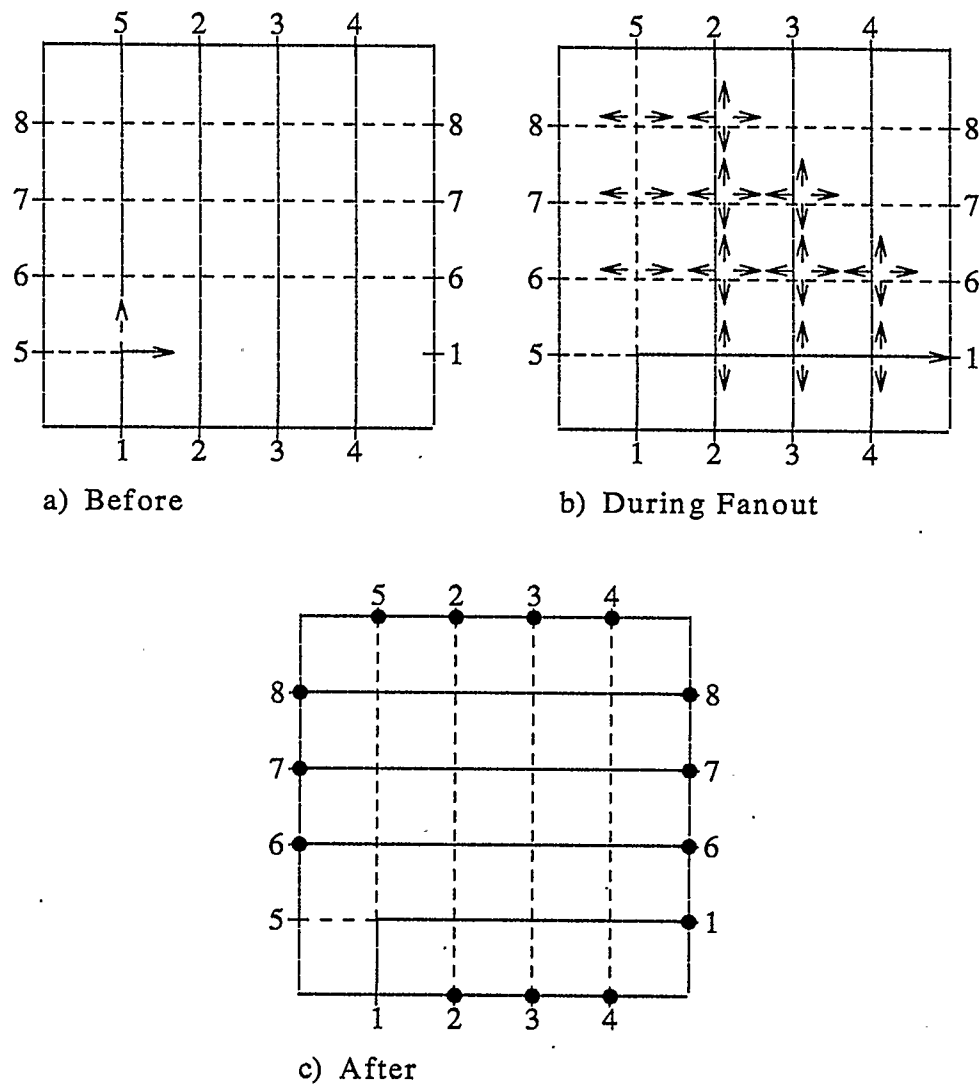


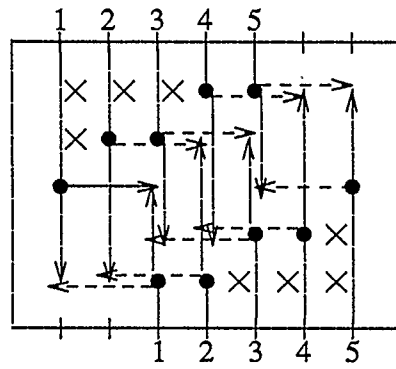
Figure 3.9 Complicated Corner Overlap Fanout Example

net expansion and constraint propagation are. Constraint propagation forces all terminals to route into the channel by one unit. This procedure is also executed for semi-convergent and divergent nets in switchbox routing. Nets are expanded, available tracks are selected, and net conflicts are resolved the same as is done in switchbox routing. Three example channel routing problems are routed in Figure 3.10 to show how these

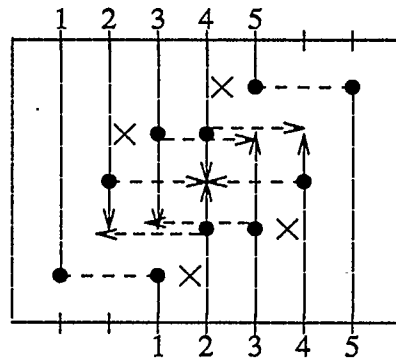
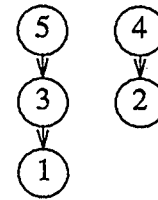
heuristics can solve this problem.

The first example is a five net channel routing problem with no overlap and therefore no cyclic constraint. Each terminal's expansion is shown in Figure 3.10a and the vertical constraint graph in Figure 3.10b. In Figure 3.10a, the upper terminals of nets 1, 2, and 3 have propagated down so that each of them can expand to the right. An 'x' marks where each net cannot find an available track to the right. The opposite propagation has been performed for the lower terminals of nets 3, 4, and 5. Once the propagation is complete, each net expands horizontally and vertically towards the other terminal in its net. The conflict resolution heuristics in this case let the vertical expansion for the furthest left terminal of net 1 route and let the furthest right terminal of net 5 route, because neither has any overlaps. The horizontal segments extending from the lower terminal of net 1 and the upper terminal of net 5 are also routed after their vertical segments are routed, because they become straight connecting routes between two hanging terminals. Straight routes are routed if nets having other than straight routes overlap with them. Once those nets have been routed, nets 2 and 4 propagate to the state shown in Figure 3.10b. Using the same track expansion techniques and conflict resolution heuristics, the problem will have the final solution shown in Figure 3.10c.

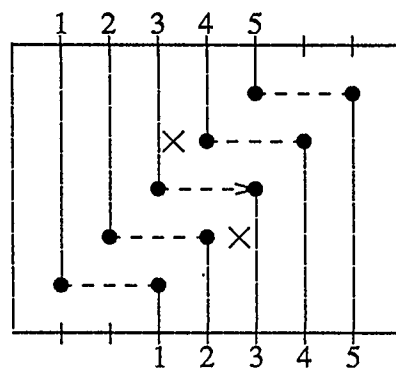
The second example shows a more complicated channel routing problem with a cyclic constraint. The terminal expansion and vertical constraint problem are shown in Figure 3.11. All routes remain blocked in this example, because the conflict resolution heuristics cannot route overlapping nets. A heuristic must choose a net from those which overlap on a similar endpoint with another net. In this case net 1 and 3 overlap on a common endpoint at column two. This indicates that only one track is



a) First Expansion



b) Second Iteration



c) Final Solution

Figure 3.10 Channel Routing Problem with no Cyclic Constraint



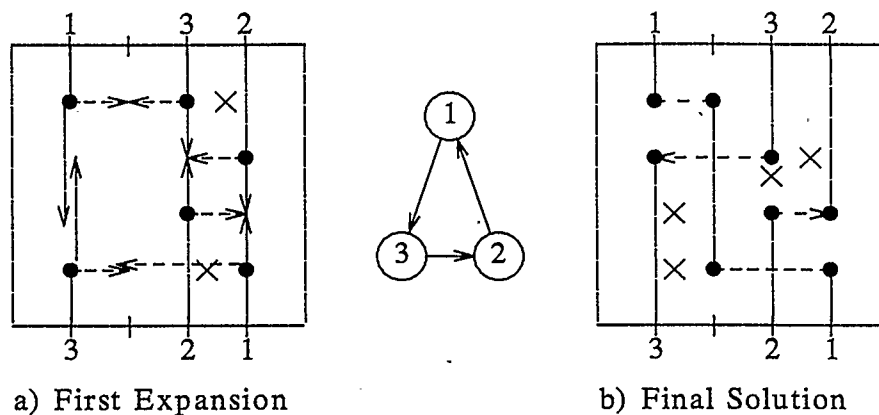
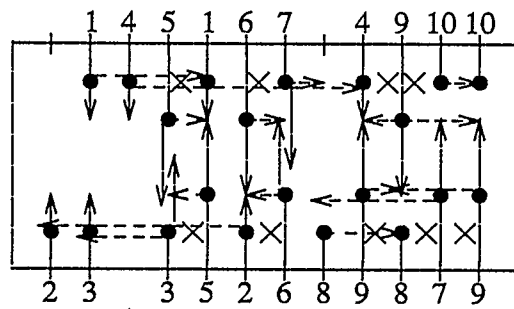


Figure 3.11 Channel Routing Problem with Cyclic Constraint

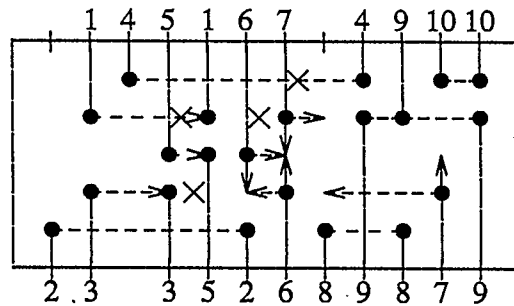
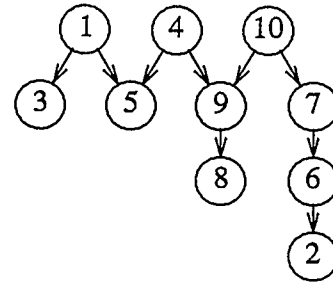
available between the two nets and the fact that they meet at this track twice (above and below) indicates that there is a cyclic constraint in the problem. Net 1 was chosen in this example arbitrarily. The next step in the solution after net 1 is routed is shown in Figure 3.11b. Nets 2 and 3 have propagated to new positions. With a broken cyclic constraint, both nets can be routed easily by the heuristics. It is important to note that picking either of nets 1 or 3 would have resulted in a simplified problem. Picking net 2 would not, because the conflict between net 1 and 3 would still persist.

The final example is a generalization of the first and shows what heuristics should be added to help route divergent nets in channel routing problems. Figure 3.12a depicts a problem with no cyclic constraint as is evident from its vertical constraint graph.

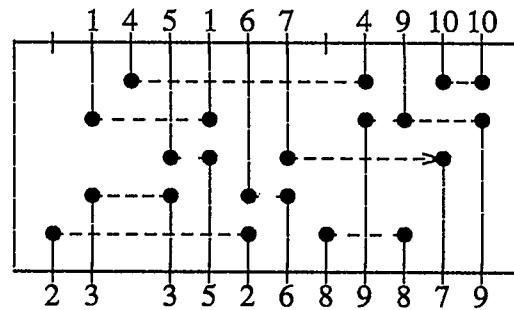
All nets are propagated past their constraints and expanded towards their other terminals. In the case of divergent nets (1,2,3,4,8,10) they only need to expand to their other hanging terminal to create a straight through route. However, because net 2 overlaps with net 3's expansion and net 1 overlaps with net 4's, their other paths of expansion should be checked to see if they can propagate in line with their routed segments



a) First Expansion



b) After One Iteration



c) Final Solution

Figure 3.12 Channel Routing Problem Requiring Net Merging

should it be required. Those nets which cannot propagate route straight across. If both can expand, one net is chosen to route arbitrarily. Once their expansions have been made, nets are selected for routing. Net 10 and net 8 route with no difficulty. Between the other divergent nets with

overlaps an arbitrary choice is made to route nets 4 and 2, thus forcing nets 3 and 1 to propagate to the next available tracks which force nets 5, 6, and 7 to also propagate to available tracks. Once net 4 is routed, the left upper corner of net 9 can route as well as its upper right corner (see Figure 3.12b).

In this figure, nets 1, 3, and 5 will route straight through. Net 6 will route the lower left corner expansions due to its conflict with net 7. Once this has been done, net 7 is left with two routing choices each. Following the heuristic established for this situation, net 7 follows its already established routing direction to the problem state shown in Figure 3.12c. Net 7 can now connect straight through to complete the solution.

From the examination of these examples, it is suggested that Marek-Sadowsa's definitions and the new heuristics created for routing switchboxes can be applied to the channel routing problem.

### 3.5. Summary

Chapter 3 presented new heuristics for routing switchboxes that allowed nets to expand vertically and horizontally and perform constraint propagation in those directions to better establish what tracks are available for routing and to select a net to route based on how its expansion interferes with other nets. Heuristics were also discussed that would allow a net to overlap the corner of another routed net if it was unable to expand towards its destination. No heuristics were investigated or presented in this thesis to perform parallel overlap or create escape routes for nets where corner overlap cannot be performed; however, the inclusion of such heuristics in a router would be useful. Finally, the heuristics for switchbox routing were examined to see how they could be used to route the channel routing problem. Chapter 4 discusses the implementation of

the heuristics for net expansion, constraint propagation, and net selection as an expert system for routing VLSI designs. Only the heuristics used to route switchboxes are not implemented, because it was felt that the success of the corner overlap and channel routing heuristics depended on the success of the new heuristics for track expansion and net selection. If these heuristics were verified to be able to solve the switchbox routing problem, then it was a trivial problem to include the other heuristics and expand the capability of the implementation to solve the general routing problem.

## **CHAPTER 4**

### **The B & D Router: An Expert System for Routing VLSI Designs**

Chapter 3 described new heuristics that are able to route switchboxes and channels by using better methods of estimated the availability of tracks and routing nets based on this estimation. Chapter 4 describes the implementation of these heuristics in an expert system for routing called the B & D router. It discusses the implementation issues present in creating an expert system to implement these heuristics, describes the general architecture and control of the B & D system, and details the basic data structures that are used to model the switchbox routing problem.

#### **4.1. The Use of Expert Systems for Routing**

Expert systems offer an suitable environment in which to implement a heuristic algorithm for good routing once the heuristics have become manifest. An expert system facilitates the modular creation of heuristics, because of its unique architecture and operation described briefly below. And expert system is composed of three distinct parts: a working memory, a knowledge base, and an inference engine. The working memory is where the problem stated is held. It contains the data structures that will represent the current problem and eventually the final solution to the problem. The knowledge base houses the heuristics. A heuristic is structured similar to an if-then statement; it is a rule containing a set of conditionals in its left hand side and actions in its right hand side. The conditionals are compared against the elements in working memory; if a match occurs, then the actions are carried out altering working memory by deleting, adding, or modifying the elements creating a new problem state. The inference engine is responsible for repeating a cycle of recognition

(matching conditionals to patterns in memory) and execution (performing the actions of rule). One cycle constitutes the firing of one rule. The system will halt when no match occurs between the conditionals of any rule and the elements in working memory or when a halt instruction is executed as the action of a rule. If more than one rule is able to fire for a given state of the working memory, the rule matching against the most specific and recent data in working memory is given precedence. This superficial discussion of a forward chaining expert system [Haye83] provides enough information to enable an explanation to be given as to why it is an appropriate medium for building a good heuristic router.

An expert system allows heuristics in the form of rules to be added independently from the control of the system. This feature gives expert systems two advantages over a conventional procedural program implementation. The first advantage is that its structure creates a highly modular and flexible system for developing heuristics. This advantage is given high marks by Steinberg [Ste84] for use in the development of knowledge-based VLSI CAD systems. In contrast, a conventional program has its if-then statements married to the control of the program and therefore heuristics cannot be developed independently from the rest of the program. The second advantage is the operation of the expert system's cycle of recognition/action which ensures that among the many heuristics present in the knowledge-base, only those that are applicable are used. In a conventional implementation, conditionals are executed as they are encountered in the code and this may or may not be when they are the most useful. These two advantages also fulfill the two requirements that an automatic router achieve the expertise on the same level as a designer. Namely, that the system be able to work with many and varied heuristics and that it have the mechanism to decide which heuristic to use and when

it was most applicable.

Thus the operation and structure of an expert system is capable of supporting the development of a good automated heuristic router. However, an expert system does have a disadvantage: slow execution speed. As the size of the knowledge base grows, so does the search through all heuristics to find those that can be applied to the current problem state, which has been noted as the limiting factor of expert systems [Part86]. To aid the search, the conditionals of the heuristics are stored as a tree of propositions against with the facts in working memory are matched. Although incremental development of heuristics improves the solutions generated from an expert system, the size of the tree will soon meet an upper bound in the number of heuristics it can sustain, because the technology of the system is inherently non-adaptive and largely inflexible, i.e. it cannot learn.

Despite this limitation, many expert systems exist which have been successful at accomplishing their tasks, whether it be analyzing protein crystallography (DENDRAL) [Benn82:106-110], solving mathematical problems (MACSYMA) [Benn82:143-149], diagnosing infectious blood diseases (MYCIN) [Benn82:184-192], or designing floorplans for the layout of computer component (R1) [McDe81]. To emphasize the success of expert systems further, R1 was developed as an expert system only after several unsuccessful attempts to develop it using conventional programming techniques. Thus theoretically and practically, an expert system implementation of a routing heuristic algorithm appears to be a good choice.

Suitable expert system shells are available that offer the development support required to produce a routing heuristic algorithm quickly and easily. Several expert system shells were investigated on their program

development support, tracing and debugging facilities, and the syntax of the programming language: ART, Inference Corporation's Automatic Reasoning Tool [ART86], OPS4 [Forg79], OPS5 [Forg81], YAPS [Alle83], and Prolog [Cloc81] [Brat86]. With regards to availability and cost, OPS4, OPS5, and YAPS are available for use by educational institutions. On the other hand, ART is only available commercially and its cost may be prohibitive for use by some organizations. However, ART was chosen over the other systems, because of its superior design environment and software development features. ART's lisp-like syntax is easier to read and understand than OPS5's and OPS4's syntax. ART also allows the data structures to be built in hierarchical fashion using a frame representation similar to that used by frame-based object oriented languages [Fike85]. It allows the user to define classifications of objects and properties that can be inherited by objects in the same class. This feature is valuable for expert systems that need to define vast numbers of elements that are similar in structure, but differ in name or in the contents of their properties. The user can define his own relationships between objects and can define his own functions for the action part of a rule, neither of which can be done in OPS4 or OPS5. Special macro expansions available in ART enable the user to write rules using case and for statements which are split into separate rules when the system is compiled to run. This valuable feature speeds up the rule-writing time and helps give structure to large and complex rules. ART is the only system in this list that has this feature. Although ART cannot support arbitrarily nested data structures as OPS4 and YAPS can, pattern matching does not have to be done on the structure of the data, just on the data itself. Its tracing capabilities are far superior to the other products, especially those offered by Prolog which is better suited to building backward-chaining expert systems. For these reasons, ART was chosen as the expert system shell for



implementing the B & D Router.

#### 4.2. The Architecture of the B & D Router

The architecture of the B & D router is based on the WEAVER's expert system architecture, which has a blackboard architecture, a system of experts that work on the blackboard sharing information and ideas for which net expansions (candidate nets) should be routed and a control expert that governs the sequence of experts that get access to the blackboard.

Similar to the WEAVER, B & D uses a blackboard architecture but differs from the WEAVER in its construction. WEAVER has three partitions of the blackboard whereas B & D has only one, the problem state partition which represents the status of the switchbox with its candidate and final routes. The second partition of the WEAVER is a scratch-pad partition where information such as vertical and horizontal constraints are kept. The third partition is the area where candidate nets are selected to route by the control expert. In the B & D system, candidate net expansions and other pertinent information is stored in the problem state representation.

B & D does not have multiple experts working on the routing problem with one expert deciding control amongst them as it is done in the WEAVER. The set of rules or heuristics work together autonomously and are applied only when the situation warrants it. It was felt that this was in keeping with the true application of heuristics in a heuristic algorithm. If one rule requires priority over another, its salience is set higher and in the event that two rules are activated simultaneously, the higher priority rule is fired. If rules of equal priority fire simultaneously, the expert system shell automatically activates the rule matching on the

most recent data.

The next section discusses the basic data structures which are used to represent the switchbox problem state in the B & D router.

#### 4.3. B & D Problem Representation

The B & D router keeps its basic data structures similar close those of the WEAVER's. Any differences that have evolved are due to the incorporation of Marek-Sadowska's definitions and heuristics into the B & D and from the removal of some WEAVER constructs that are not useful to the B & D approach to switchbox routing.

Two input structures that are accepted by B & D are the channel (see Figure 4.1) and the pin (see Figure 4.2). The channel is represented by the four coordinates of minimum and maximum x and y coordinates which establish rectilinear channel boundaries.

Similar to WEAVER in the way it created the horizontal and vertical routing wires at each column and row, B & D has relations defining column and row numbers (see Figure 4.2) are created for each routing position available in the channel between the minimum and maximum x and y values. The columns and rows are positioned one unit apart in accordance with the grid approach to routing.

The second data structure accepted as input into B & D is the pin structure. B & D defines several categories of pins as indicated by the

```
(defschema channel
  (min-x)
  (min-y)
  (max-x)
  (max-y))
```

Figure 4.1 Channel Schema Definitions

```
(defrelation column-number
  (instance-of relation))
```

```
(defrelation row-number
  (instance-of relation))
```

Figure 4.2 Row and Column Relations

'kinds' relation in the example definition below in Figure 4.3.

The inheritance facility of ART is implemented by using 'kinds'. It instructs the system that if any structure is built of this kind, it is to also contain the default slots of its super structure. In the above example, pin-1, a hanging pin, inherits all the default slots of the master type 'pin'. All pins that are input to B & D are defined as terminal pins. Their x and y coordinates are on the switchbox's boundaries. Hanging pins and corner pins are defined to be at the end of straight and corner expansions as per Marek-Sadowska's definitions. A routed pin is created when all connections emanating from a pin have been routed. Terminal pins can be routed pins, but corner and hanging pins cannot because they are used as points of expansion and routed pins are not. Terminals are kept as markers to aid in determining to which net classification an expansion route belongs. For example, a two terminal net with linear boundary coordinates indicates that the net is divergent. The pin-in-net slot is a relation slot which creates a link between the pin and the net in which it

(defschema pin	(defschema pin-1
(pin-in-net)	(instance-of hanging-pin))
(pin-x)	automatically added to pin-1:
(pin-y)	(instance-of pin)
(kinds terminal-pin	(pin-in-net)
hanging-pin	(pin-x)
corner-pin	(pin-y)
routed-pin))	

Figure 4.3 Pin Schema Definitions

belongs. For the net definition in Figure 4.4, the opposite relationship is established by using the '-has-' construct.

A connection is defined as the construct that joins two pins in a straight line segment (see Figure 4.5). Two classifications are created to discern candidate expansions from their routed counterparts, and the third classification defines the connections that make up the boundary of the channel. Rather than describing connections as being 'convergent', 'semi-convergent', and 'divergent', connections are classified by pin types and the way they intersect. For example, the connections from two pins are considered to be a corner route if the two connections are perpendicular to each other and they meet on a common point of the type 'corner pin'. Hanging pins are the endpoints of routed connections that have no other routed connections emanating from them. A special 'cannot-expand' slot is defined to handle the cases where a corner pin or hanging pin cannot expand in a certain direction.

The representation of routed segments by the connection construct overcomes a problem that Joobbani faced in his implementation. He kept track of the wires available for routing by using two data structures that

```
(defschema net
  (net-has-pin)
  (net-is-routed no))
```

Figure 4.4 Net Schema Definition

```
(defschema connection
  (connection-has-pin)
  (connection-has-pin)
  (kinds candidate-connection
          routed-connection
          channel-connection))
```

Figure 4.5 Connection Schema Definition

represent the horizontal and vertical wires which extend the entire length and width of the switchbox. Two sets of the structures are created, one for each routing layer. As parts of a wire are routed for one layer, the parts are removed from consideration by deleting the length of wire unavailable for routing in the same layer. He cites that it would be advantageous to represent each individual segment of the switchbox grid as separate routing wires, but the size of some routing problems and the number of data structures required makes it infeasible to do this properly.

B & D's representation does not keep track of available routing area; instead it keeps track of wires as they become routed, i.e. as they change status from candidate-connection to routed-connection. Before channel routing begins all tracks and columns are available, because no routed connections exist. Pre-routed nets can also be included in the system by manually asserting their path as segments of routed connections.

There are no higher level structures built on top of the basic structures just outlined. For example, corner routes are recognized by comparing two connections to see if they meet in a corner. No special structure is asserted to state the existence of such a corner route. In an expert system the extra matching required to recognize graphical information such as this can be great. But the state of a routing problem changes rapidly as each partial net is routed and many candidate connections are asserted and deleted before they are finally routed. The extra time required to maintain the higher level structures would make it a burden rather than an aid to keep track of them.

#### **4.4. The B & D Program**

The B & D program comprises all the heuristics discussed in chapter 3 and are outlined below in Table 4.1. As well it includes heuristics to

initialize the channel, create columns and rows, and define input pin locations.

Approximately 3900 lines of ART Code were developed to create these rules Figure 4.6 shows a sample ART rule to perform constraint

	Written -----	Compiled -----
Define Channels, Nets, and Pins	3	6
Create Columns and Rows	3	6
Generate Corner Segments	4	72
Identify Overlap Between Corner Segments	1	26
Route Semi-Convergent and Divergent Nets (1 unit)	1	13
Propagate Constraints Straight Forward	1	50
Propagate Constraints Around Corners	2	8
Expand / Can't Expand Hanging Pins	4	92
Route Non-Overlapping / Straight Connections	1	14
Route One of Two Choices	1	9
Delete Overlaps	1	4
Delete Hanging / Routed Pins On Parallel Segments	3	9
Corner Pins to Hanging Pins	2	15
Corner Pins to Routed Pins	1	1
Miscellaneous	3	3
TOTAL	----- 31	----- 370

Table 4.1 Summary of the Rules Written for B & D

propagation where a hanging pin intersects a routed segment. Compared the size of the WEAVER in the number of rules, the B & D router is small. WEAVER has over 700 rules as opposed to 31 in B & D. However, ART's program development features allow a rule to be created with far less code than one for OPS5, which was Joobbani's choice of implementation language. When ART compiles the rules, they expand into the actual number used by the expert system. The number of compiled rules in B & D is shown above. Thus one ART rule is equivalent to several written in OPS5. However, size is not a true estimate of quality; the output is and this is discussed in chapter 5.

#### 4.5. Disadvantage of Expert Systems

Chapter 1 described at length that an expert system is a good implementation medium for representing and solving difficult problems; however, programming an expert system can be difficult.

An expert system's main disadvantage is its inability to do anything procedurally in a straight-forward manner. For example, if a program is used to add the numbers from one to ten (see Figure 4.7), a simple `for` loop with an increasing sum will suffice. However, this is more complicated in an expert system. First the sum must be placed in working memory as a fact. The rule to add it must pattern match on the sum, check if it is less than or equal to 10, and then a), delete the old sum from working memory b), execute a routine to add sum to itself plus 1 and c), place the new sum in working memory again. The structure of an expert system obscures the procedural flavour of some code. Most expert system applications however, and this is true for B & D, are not designed to do procedural work. They were designed to implement heuristic algorithms, most of which involve symbol manipulation and pattern matching, not

```

;;;DEFRULE CP-TO-GET-A-TRACK

;;;This rule extends hanging pins beyond an already routed track
      (defrule cp-to-get-track
        (declare (salience ?*default-salience*)))

;;;Match on a hanging pin
      (schema ?hanging-pin
        (instance-of hanging-pin)
        (pin-in-connection ?connection)
        (pin-in-net ?net)
        (pin-x ?pin-x)
        (pin-y ?pin-y))

;;;Make sure it is a hanging pin i.e. no other connections extend from
;;;it except the one above
      (not (exists (pin-in-connection ?hanging-pin ~?connection)))

;;;Match on information of the hanging pin's connection
      (schema ?connection
        (instance-of routed-connection)
        (connection-has-pin ?other-pin &~?hanging-pin))

;;;If the hanging pins only connection is vertical, find another connection
;;;that crosses its endpoint horizontally
      (split ((connection-dir ?connection vert)
        (schema ?other-connection
          (instance-of routed-connection)
          (connection-dir horiz)
          (connection-has-pin ?pin-1)
          (connection-has-pin ?pin-2 &~?pin-1)))

;;;For this other connection to cross on the hanging pin's endpoint
;;;check the other connection's pin-x values to make sure they surround
;;;the endpoint and overlap with it on its pin-y value
      (schema ?pin-1
        (pin-in-net ~?net)
        (pin-x ?pin-x1 &:(?pin-x1 <= ?pin-x))
        (pin-y ?pin-y))
      (schema ?pin-2
        (pin-x ?pin-x2 &:(?pin-x2 >= ?pin-x)
          &:(?pin-x2 > ?pin-x1))
        (pin-y ?pin-y))

Continued on the next page...

```

Figure 4.6 Example ART Rule



```

;;;Finally, if the hanging connection's other pin is higher than the
;;;hanging pin, then decrease the hanging pin's pin-y value so that
;;;in effect the hanging pin is extending downwards

```

```

      (split ((pin-y ?other-pin ?higher-y
                  &:(?higher-y > ?pin-y))
              =>
              (modify (pin-y ?hanging-pin =(- ?pin-y 1))))

```

```

;;;Otherwise, if the other pin is lower, increase the hanging pin's pin-y
;;;value to that its hanging connection will extend upwards

```

```

      ((pin-y ?other-pin ?lower-y
              &:(?lower-y < ?pin-y))
       =>
       (modify (pin-y ?hanging-pin =(+ ?pin-y 1))))

```

```

;;;SECOND HALF OF FIRST SPLIT

```

```

;;;The second half does the same as the first, except now the hanging
;;;connection is a horizontal connection, and another routed connection
;;;may overlap it in the vertical direction

```

```

      ((connection-dir ?connection horiz)
       (schema ?other-connection
                (instance-of routed-connection)
                (connection-dir horiz)
                (connection-has-pin ?pin-1)
                (connection-has-pin ?pin-2 &~?pin-1))
       (schema ?pin-1
                (pin-in-net ~?net)
                (pin-y ?pin-y1 &:(?pin-y1 <= ?pin-y))
                (pin-x ?pin-x))
       (schema ?pin-2
                (pin-y ?pin-y2 &:(?pin-y2 >= ?pin-y)
                    &:(?pin-y2 > ?pin-y1))
                (pin-x ?pin-x))
       (split ((pin-x ?other-pin ?higher-x
                   &:(?higher-x > ?pin-x))
               =>
               (modify (pin-x ?hanging-pin =(- ?pin-x 1))))
               ((pin-x ?other-pin ?lower-x
                   &:(?lower-x < ?pin-x))
               =>
               (modify (pin-x ?hanging-pin =(+ ?pin-x 1))))))
    )

```

```

;;;END OF DEFRULE CP-TO-GET-A-TRACK

```

Figure 4.6 Continued

int sum;	(defrelation (sum (?sum))
sum = 0	(deffact (sum 0))
	(defrule rule-1
for i = 1 to 10 do	?x <-(sum ?sum-num
	&:(?sum-num <= 10))
	=>
sum = sum + i	(retract ?x)
	(assert (sum (+ ?sum-num 1)))
Procedural Code	ART rule

Figure 4.7 Procedural Versus ART Code

procedural calculations.

One other disadvantage which is a direct result of the modularity and flexibility of expert systems is the ease to which ad hoc heuristics can be added to the system. As the system grows, the addition of these many and varied heuristics makes it more difficult to track what the system is doing and why. Therefore, it is important to know where newly created heuristics fit into the design and operation of the system before they are added, because it is difficult to find out what will happen after the fact.

#### 4.6. Summary

Chapter 4 discussed the general architecture of the B & D routing expert system, its data structures, and a synopsis of the rules included in the system to perform the heuristics described in chapter 3. Chapter 5 presents the experimental results of the implementation on several switchbox and channel routing problems and compares the results to those of other routing algorithms.

## CHAPTER 5

### Experiments with the B & D Router

This chapter summarizes the experiments done with the B & D router. The first section shows the input and output configuration used in each of the experiments conducted on B & D. The second section steps through an example problem and describes the heuristics used to route each step of the example. The third section lists the examples and the results obtained by B & D and compares these with the WEAVER's, Marek-Sadowska's, and other system's results on Burstein's Difficult Switchbox Problem. The final section discusses how well B & D does overall as a router and where improvements can be made.

#### 5.1. Input and Output

The input and output configuration of the B & D program is modeled after the form used in Joobbani's WEAVER. A simple fact list of channel and terminal specifications suffices for input; a partial fact list for the switchbox problem in Figure 5.3 is shown in Figure 5.1. The channel's minimum and maximum x and y coordinates are all that are necessary to define it. A terminal's location requires its name x and y coordinates, the net it belongs to, the layer on which it is to be routed, and the side of the channel it is located on.

The output format was designed to be similar to Joobbani's and is represented by facts in the working memory of the exert system. A partial list of B & D output facts for net 5 in Figure 5.3 is shown below in Figure 5.2. A new name for a connection pin is created by merging its net name along with the new x and y coordinates of the pin. Connection

```
(channel-input ?min-x ?min-y ?max-x ?max-y)
(pin-input ?pin-name ?net-name ?pin-x ?pin-y ?channel-side)
```

a) Input Facts Form

```
(channel-input 0 0 13 8)

(pin-input pin-1 net-2 1 0 down)
(pin-input pin-2 net-3 2 0 down)
(pin-input pin-3 net-5 3 0 down)
(pin-input pin-13 net-9 13 1 right)
(pin-input pin-18 net-6 0 1 left)
(pin-input pin-19 net-5 0 2 left)
(pin-input pin-22 net-1 2 8 up)
```

b) Partial Facts List for Problem in Figure 5.3.

Figure 5.1 B & D Input Example

```
(schema net-5-3-2-pin-2-conn
  (instance-of routed-connection)
  (connection-has-pin pin-2)
  (connection-has-pin net-5-3-2)
  (connection-dir vert))

(schema net-5-3-2
  (instance-of routed-pin)
  (pin-in-net net-5)
  (pin-x 3)
  (pin-y 2)
  (pin-in-connection net-5-3-2-pin-2-conn)
  (pin-in-connection net-5-3-2-pin-19-conn)
  (pin-in-connection net-5-3-2-net-5-5-3-conn)
  (pin-in-connection net-5-3-2-net-5-3-5-conn))
```

Figure 5.2 B & D Example Output

names are established by concatenating the names of the two pins together and appending 'conn'. Although this output is cryptic, it does serve the basic purpose of showing where nets route. Building a more spectacular interface was not worthwhile for a prototype system. However, for the remainder of this chapter all output has been displayed graphically to

support the reader in his endeavours to understand this material. The use of arrows, lines, and bullets is the same as it was used in Figure 3.6.

## 5.2. An Example Run with B & D

Figure 5.3 displays an example switchbox problem, which has been solved by the B & D router. It is the aim of this section to show how the heuristics are applied step by step to solve the example. Figure 5.3 shows the example problem after corner routes have been generated for the problem. A second heuristic has routed all semi-convergent and divergent routes into the channel by one unit. Two conflicts arise between net 8 and net 9, and between net 2 and net 6. Because nets 2 and 8 have been routed up to the point of conflict, the upper terminals of nets 9 and 6 will be reclassified as hanging terminals and their routes pushed back. Figure 5.4 shows the resolution of the conflicts in Figure 5.3.

The first non-conflicting corner routes are now routed in Figure 5.4 and now expand to meet other terminal extensions as nets 1, 3, 5, 7, and 9

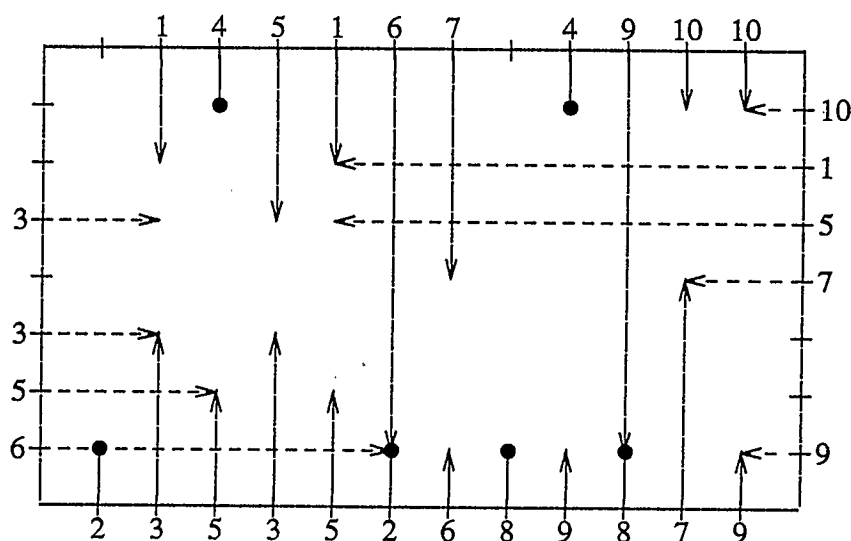


Figure 5.3 First Step of an Example Switchbox Routing Problem

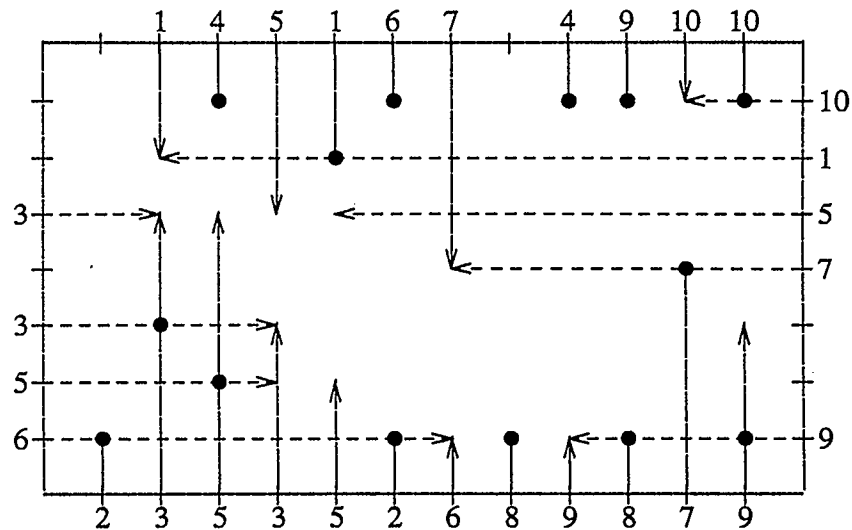


Figure 5.4 Step Two

demonstrate. All hanging terminals have been routed to at least as far as the first unit within the switchbox boundary; they will not expand further until all corner routes and their extensions have been completed.

In Figure 5.5, all corners and corner extensions that have no conflicts have been routed. However, constraint propagation has taken effect forcing the lower terminals of nets 2 and 8 up to an available track. Now that all corners have been routed, hanging terminals can expand using the track availability heuristics. This includes the divergent nets 2, 4, and 8, and the semi-convergent nets 9 and 6.

As is explained in chapter 3, a net's terminals are expanded towards each other by finding the furthest track available before it meets the routes of other terminals. To graphically depict that a hanging terminal cannot expand in a certain direction, an 'X' will be placed next to the terminal in that direction. Candidate routes are prohibited from expanding back across routed connections to ensure that routes will expand from hanging

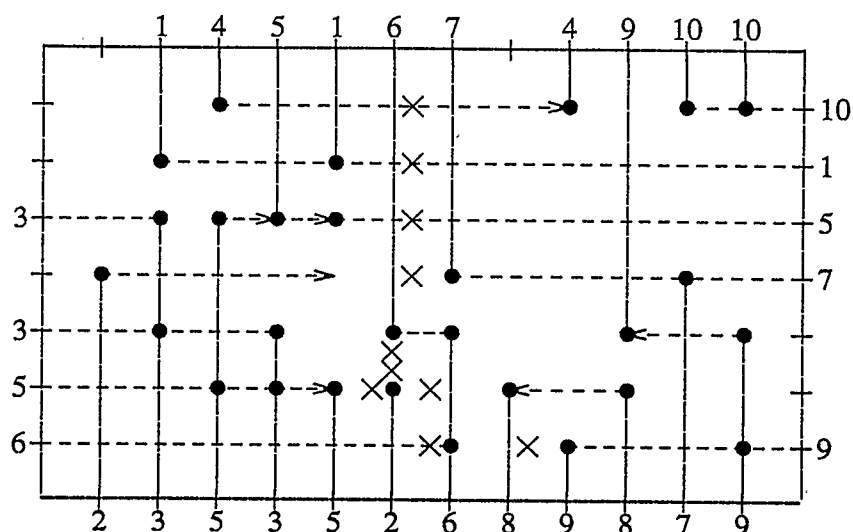


Figure 5.5 Step Three

pins only, not from the middle of routed segments. Figure 5.5 shows the expansions made from all hanging terminals.

Corner routes are considered hanging terminals if they are not as yet connected to the other terminals. This presents a problem if there is more than one corner from which an extension can be made. In B & D, a corner is disallowed from expanding in a direction to connect to another terminal if another corner expansion already spans the same area. Net 9 in this example demonstrates this phenomenon. There are two corners from which an expansion could be made towards the upper terminal. Since both span the same area of the switchbox, only one is finally considered as a candidate route. No heuristics are used to decide which corner should be allowed to expand; the choice is made arbitrarily.

Figure 5.6 shows the result of applying the net conflict resolution heuristics to the expansions made in Figure 5.5. Because the upper terminal of net 6 could not expand in the one direction and its other

expansion has a conflict, it has propagated downwards. The same has happened to both terminals of net 2. The lower terminal on net 6 cannot expand in one direction, but its other expansion has no conflict so it is able to route it to completion. The same is true for net 9. All other expansions with no overlap are also completed. This includes the expansions connecting nets 4, 5, 8, and the expansion from the lower terminal of net 9.

By Figure 5.6, all nets have now been routed except for net 9, which has an easy solution, and nets 2 and 6 which have a difficult solution. Net 9 has a straight through expansion with no overlap which can route. Net 6 becomes constrained to propagate past nets 2 and 7 and becomes interconnected. Net 2 whose two terminals have only one non-overlapping expansion each can also route. Net 6, on the other hand, cannot expand in either direction. Its only choice is to overlap on the corner of an already routed net. Its choices out of this predicament are to overlap on either net

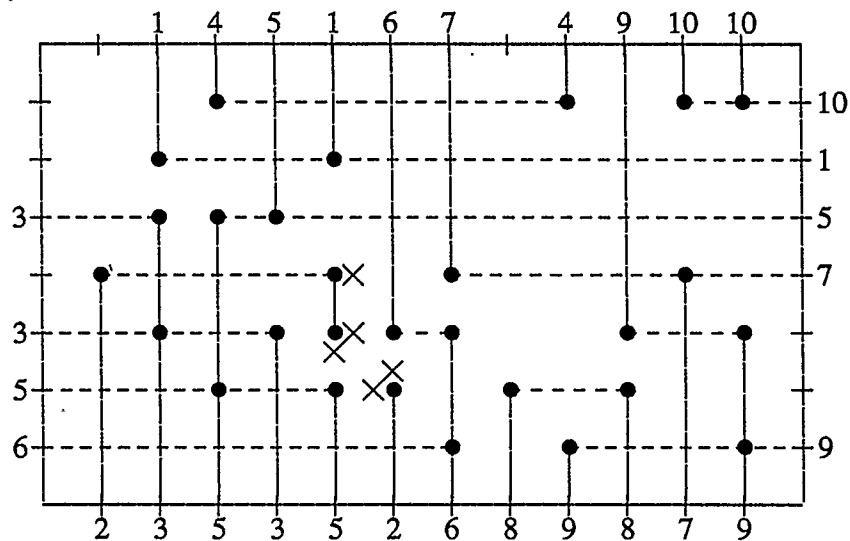


Figure 5.6 Step Four



2 or net 7; net 7 is chosen arbitrarily.

This example ran on a Symbolics Lisp Machine in 1408.15 seconds and executed 1835 rules at an average firing time of 1.30 rules per second. More examples were run on B & D and the results obtained are discussed in the next section although not in as great detail as for this example.

### **5.3. B & D Against Other Switchbox Routers**

In this section, B & D's output is compared with the output of other switchbox routers on some difficult problems to demonstrate the worthiness of the track availability, net-conflict resolution, and corner overlap fanout heuristic at obtaining good routing solutions. Two relatively easy and three difficult examples are compared. The switchbox solutions for each problem will be shown for each system and a table of statistics follows each example problem. The statistics that were taken compare the systems on their quality of the solutions they generated and quality is measured by the amount of wire used to route the problem and the number of vias required to change layers. These are superficial measurements of quality and in no way describe the difficulty of the problem being routed. After these results, B & D are compared against the WEAVER in execution speed and the number of rules fired. Again, this is a superficial comparison, but does give an idea of where each expert system stands in its ability to solve the difficult switchbox routing problem.

#### **5.3.1. A Simple Switchbox**

Figure 5.7a below shows the WEAVER's solution to the first example problem. B & D's solution is shown in Figure 5.8b. The statistics for example 5.7 are encapsulated in Table 5.1 below. Although B & D obtained a different layout than did WEAVER, the wire lengths are the

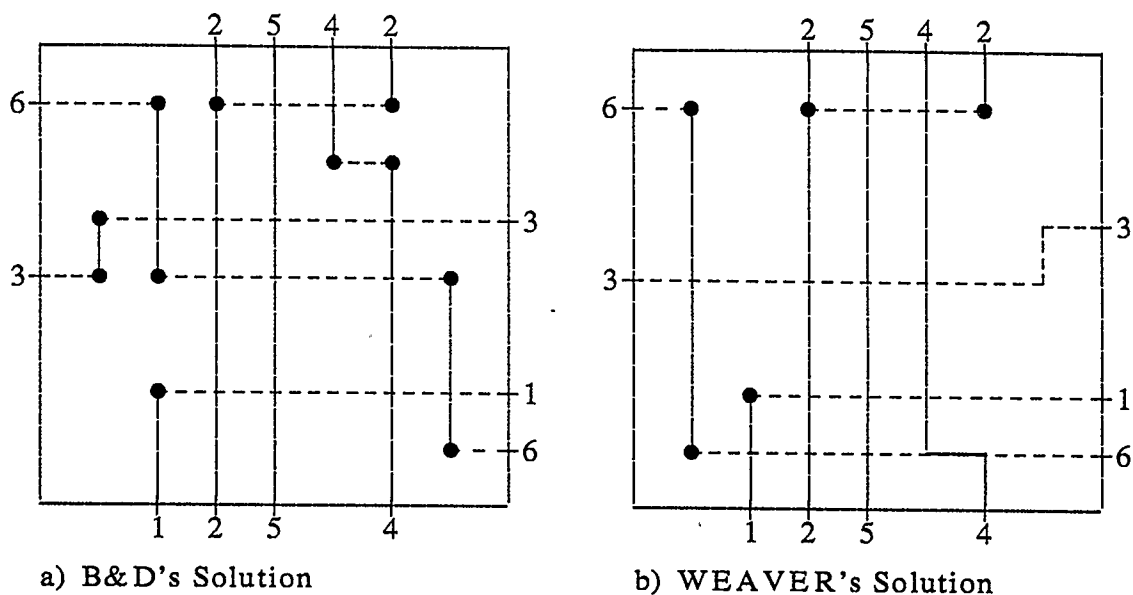


Figure 5.7 Solutions to a Simple Switchbox Problem

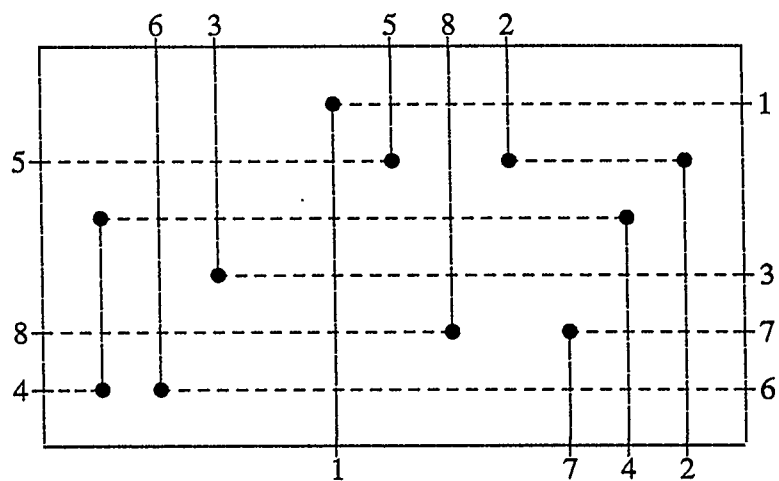
Simple Switchbox Statistics		
System	Wire Length	No. of Vias
WEAVER	60	4
B & D	60	11

Table 5.1 Statistics for the Simple Switchbox in Figure 5.7

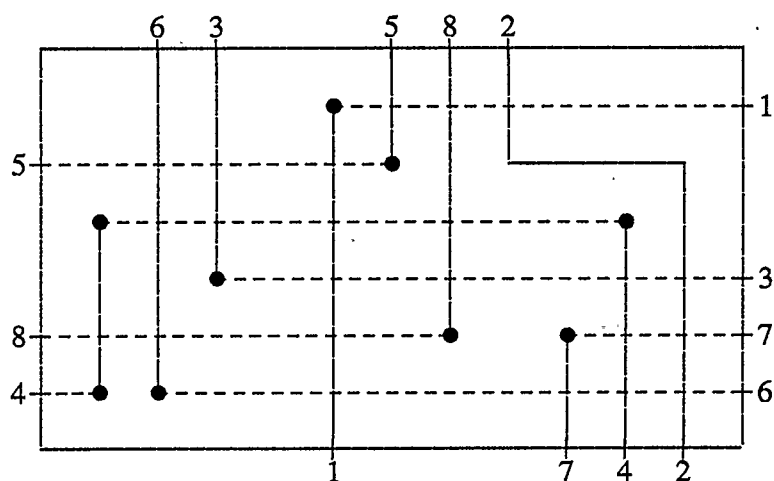
same. B & D's major drawback is in the number of vias that were required to route the solution, because it uses the standard two-direction-two-layer wiring model. The WEAVER, on the other hand, has specialized heuristics that minimize vias in routes after a net is interconnected with impressive results. However, it should be noted that WEAVER allowed net 4 to overlap net 6, which is not necessary to complete a solution.

### 5.3.2. A Second Simple Switchbox

Figure 5.8a below shows the WEAVER's solution to a second switchbox problem that can be solved with no difficulty. B & D's solution is shown in Figure 5.8b. The statistics for example 5.8 are given in Table 5.2. Here B & D again compares well against WEAVER in the wire



a) B&D's Solution



b) WEAVER's Solution

Figure 5.8 Solutions to a Second Simple Switchbox Problem

Second Simple Switchbox Statistics		
System	Wire Length	No. of Vias
WEAVER	94	9
B & D	94	11

Table 5.2 Statistics for the Second Simple Switchbox

length and layout although it requires two more vias to route the solution.

### 5.3.3. Burstein's Difficult Problem

The third comparison is between B & D's and other systems' solutions to Burstein's Difficult Switchbox Routing Problem. While the other examples were given to show how B & D compared with the WEAVER in problems that are easily solved by automated routers, Burstein's Difficult Switchbox Routing Problem was selected because it is a good test of an automated router's ability to solve the switchbox routing problem. Five other systems besides B & D have attempted the solution and some have succeeded. Burstein's solution was printed in Figure 2.15, Hamachi's (MAGIC) in Figure 2.16, Marek-Sadowska's in Figure 2.20, and Joobbani's (WEAVER) solution in Figure 2.22. Luk's answer to the switchbox routing problem was included by Joobbani in his thesis, because its solution was comparable to Joobbani's and is re-printed here in Figure 5.9.

Luk's router is an implementation of the greedy switchbox router similar to Hamachi's. B & D's results are shown in Figure 5.10.

The table of statistics for these five solutions is shown in Table 5.3. It is mentioned again at this point that Hamachi's column sweep router and

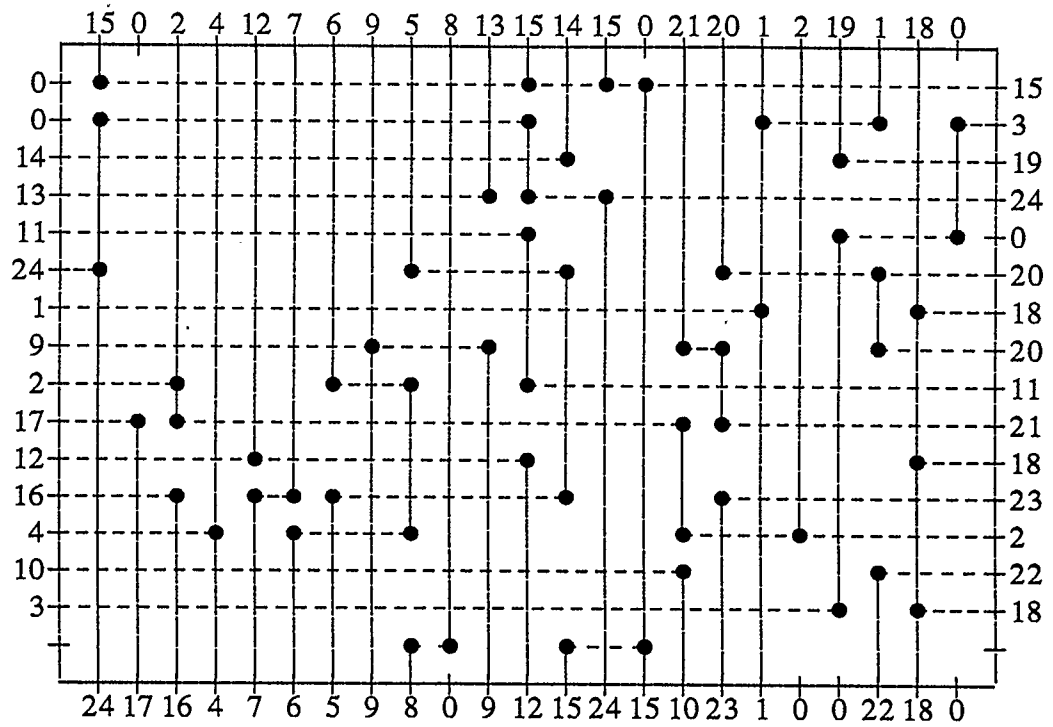


Figure 5.9 Luk's Solution to Burstein's Problem

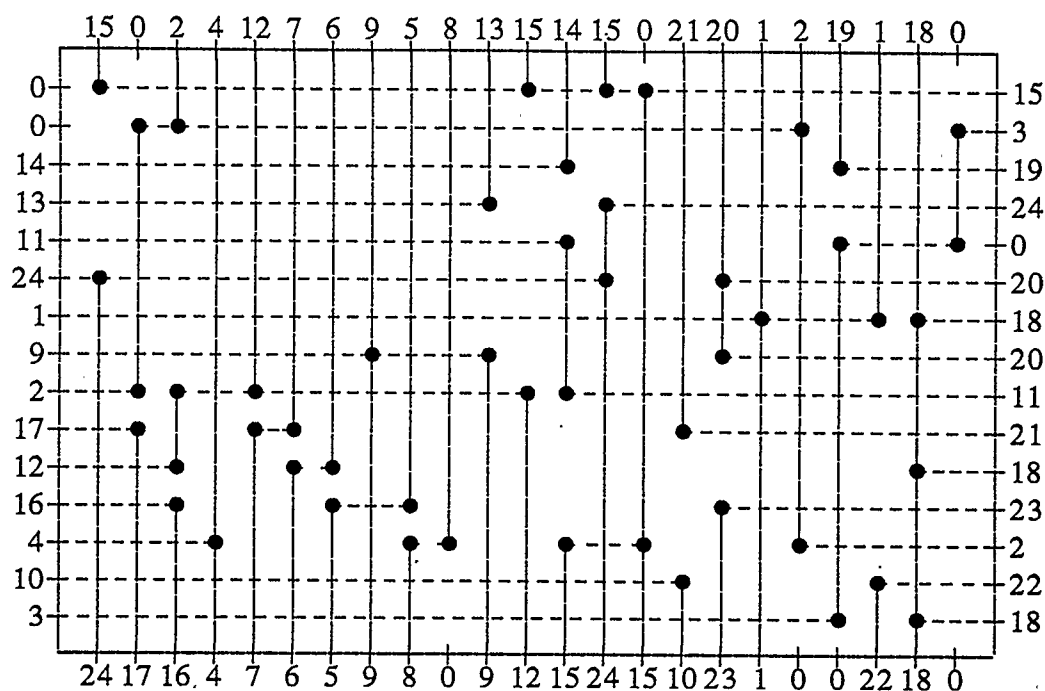


Figure 5.10 B &amp; D's Solution to Burstein's Problem

Burstein's Switchbox Statistics			
System	Wire Length	No. of Vias	
WEAVER	531	41	
MAGIC	564	67	
Burstein's	486	51	
Marek-Sadowska's	559	58	
Luk's	577	58	
B & D	536	53	

Table 5.3 Statistics for Burstein's Difficult Switchbox

Burstein's pattern router were unable to complete the entire problem. Luk's version of the difficult problem added an extra row to its definition, possibly to aid the switchbox router in completing its solution. From these results it can be said that B & D is a good router for performing switchbox

routing and its results in wire length and number of vias is comparable to WEAVER's, the system which has obtained the best results to date of other switchbox routers.

#### 5.4. B & D's Comparison Against the WEAVER

The last section in this chapter analyzes the performance of B & D with the WEAVER. This comparison is made to show how well B & D did as an expert system against another. The WEAVER is well known in the switchbox routing field as a ground-breaking and competent system, which has solved many routing problems better than other systems have in the past. B & D was compared against the WEAVER in the quality of its solutions, in this section it is compared against the WEAVER in execution speed and the number of rule firings. The five example runs were used as the basis of comparisons. The results of the statistics are given below in Table 5.4. From the comparison, it is fairly obvious that the WEAVER runs much faster than B & D in all three cases and uses less rules to find solutions. There are several reasons why this is. First, Joobbani

Execution Statistics				
Example	System	Running Time (sec.)	Rules Fired	Ave. Rule Firings (per sec.)
Figure 5.7	WEAVER	73	368	5.00
	B & D	254	995	3.91
Figure 5.8	WEAVER	151	628	4.16
	B & D	383	464	1.21
Figure 5.9/10	WEAVER	1508	3624	2.43
	B & D	36580	4896	0.13

Table 5.4 Execution Statistics of B & D and the WEAVER

optimized the WEAVER's code to run faster; no optimization was performed on B & D. Second, it is possible that the premises of the WEAVER's rules are much shorter, for example having an average of 5 conditionals, than those of B & D's which average 20. Third, it may be that OPS5 is a faster expert system shell than ART; the average rule firings per second indicate this. ART is an expert system that provides the user with an excellent software development package and tracing facilities, but its high overhead slows its execution speed. Its performance seriously degrades as the difficulty of the routing problems become more difficult, which is only to be expected because the amount of information being asserted and deleted from working memory is larger for the more difficult and larger routing problems. Thus B & D can be made to run faster by decreasing the number of conditions in the premises of its rules, by optimizing how the rules are written, by re-implementing the rules on a faster expert system shell which has a low overhead.

### 5.5. Summary

In chapter 5, B & D was run on four example switchbox problems. The first showed how the heuristics in B & D were applied to route the problem. The second and third examples were selected to obtain results of the B & D router against Joobbani's WEAVER. Burstein's Difficult Switchbox Routing Problem was selected as the fourth example to demonstrate how well B & D could do at routing this problem. From the experimental results using B & D, its heuristics for routing switchboxes are comparable to other systems performing the same tasks. Its expert system implementation is slower than the WEAVER at obtaining solutions, but this can be improved by using the techniques described above. Chapter 6 concludes thesis and makes suggestions for future work based on the results obtained by B & D.



## CHAPTER 6

### Conclusions and Future Work

Routing is an important and time consuming task in VLSI design for which automated routers try to attain the same quality as human experts. Due to its inherent difficulty, heuristic algorithms are used to solve the routing problem. Various models of the routing problem use different heuristics to find good solutions owing to difference in the physical constraints placed in the models. Current channel routing approaches take advantage of their ability to expand in height and use vertical and horizontal constraints, net length, channel density, and net merging as constraints. Switchboxes, which cannot change in size, are routed by net expansion, constraint propagation, and corner filling heuristics. Within switchbox routing, different heuristics are used to resolve conflicts between nets. General channel routing methods and heuristics from switchbox routing to perform net expansion and constraint propagation are used to resolve conflicts and help achieve a 100 per cent completion rate for all routing problems. By the inspection of the current routing approaches, it was discovered that the net expansion and conflict resolution heuristics used for switchbox routing were too limited in certain cases to achieve 100 per cent routing completion and needed to be enhanced. Secondly, although the channel routing model is similar to the switchbox model and two different sets of heuristics have been able to route the switchbox routing problem, no one set of constraints currently exists that can route both switchbox and channel routing problems.

To accomplish the first directive and the second directive new heuristics were presented that enhanced Marek-Sadowska's current

constraint propagation and net conflict resolution heuristics. They allowed nets to expand in two directions instead of one, and selected routes based on what tracks were available for each net. As well, corner overlap heuristics were presented that covered the case where nets could not expand in either direction. These switchbox heuristics were discussed as to how they could be applied to the channel routing problem. The heuristics were implemented as an expert system in the B & D router. They were tested against two simple switchbox routing problems and Burstein's Difficult Switchbox Routing Problems and achieved results comparable to the results of current switchbox routing approaches.

### 6.1. Conclusions

The heuristics developed as an enhancement to Marek-Sadowska's constraint propagation, net expansion, conflict resolution heuristics obtain good switchbox routing solutions. By including the heuristics that perform corner overlap, the router will be able to complete routes that otherwise would not be finished using the two-direction-two-layer wiring model. To perform channel routing, heuristics to choose between nets which overlap on a single endpoint and choose between overlapping divergent nets would have to be added. As was discussed in chapter 3, the first heuristic can be used to break the cyclic constraint present in a channel routing problem.

The expert system is a good medium for building a prototype system to solve the difficult routing problem because of the ease with which heuristics can be added to its flexible and modular system. However, speed prohibits its use on large problems, problems involving on the order of 10,000 nets. Solutions to this problem are to optimize the current rules so that they execute faster, recode the rules to work on a faster expert system shell, recode the system using a procedural language, or create an ASIC (Application Specific Integrated Chip) or VLSI chip whose

architecture has been developed to specifically run expert system code thereby gaining a great advantage in speed along with the good heuristics for solving the routing problem.

The system as it stands can route the standard model of the routing problem using the grid approach and the two-direction-two-layer routing model. This restricts it from use in full custom or gate array VLSI design. However, the system can be upgraded to work in a full custom environment by modifying heuristics to allow wires to route according to design rules instead of at a unit distance. Wire definitions can also be altered to define wires of different widths; presently wires have no dimension in width. The system can also be modified to work in a gate array design environment by defining the presence of pre-masked routing layers and their location in the system. Heuristics can be added route the second layer knowing the location of the first. Thus it is not difficult to modify the router so that it can route within and by the constraints of a specific VLSI design environment.

## **6.2. Future Work**

Additional work needs to be done in developing heuristics that find escape routes for nets away from conflicts if corner overlap or parallel overlap is the only alternative. These heuristics could be based on line propagation heuristics for finding escape routes for nets when they are confronted by an obstacle. The solution could also entail expanding the constraint propagation heuristics, to allow terminals to propagate to an available track (if required) in any direction regardless if it is moving towards or away from its destination.

Because the heuristics in the B & D router can route channels, as well as switchboxes, the heuristics may be of use for routing three-sided

channels. Three-sided channels, as they are defined in chapter 2, are less constrained than switchboxes, because of the presence of floating terminals on one side of the channel. Some VLSI design systems define routing areas using this model. The net expansion and constraint heuristics could be used to route the expansions for the fixed terminals away from their three boundaries towards the fourth boundary containing the floating terminals. The positions of the floating terminals would be established by noting what tracks were available for the net expansions as they proceeded across the channel.

Another interesting area for future research is in the incorporation of some of the routing heuristics in with placement routines to help these routing better judge where to place cells based on the estimate of routing space required for its interconnection. Currently, channel density is the measurement used to gauge how much area is required is not always accurate. Although, not all heuristics could be incorporated in the placement routines, a subset of them may provide a more accurate guide than is presently available.

The final area of future work deals with the implementation of the switchbox routing heuristics in a backwards chaining expert system or in a logic programming language such as PROLOG. Using the heuristics as definitions by which to attain good switchbox routing solutions, the backward chaining system is the more natural form of implementation for this kind of problem. These systems are better able to match on higher level structures and break it into its lower level components. This ability would be useful for defining the higher level structures in routing such as corner routes which currently are not structured in the forward chaining system and must be recognized by the arrangement of individual components.

### 6.3. Concluding Remarks

An expert system router for solving the general routing problem has been presented in this thesis. Building upon switchbox routing heuristics to perform net expansion and constraint propagation, new heuristics have been developed to allow nets to expand in two directions and resolve conflicts based on how able a net is to expand to its available track in either direction. Although the constraints do not ensure 100% completion, they do emphasize the importance of it in switchbox routing problems by ensuring that nets which are the most constrained in their choice of available tracks get priority in selecting their routes.

## References

- [Abu87] Abu-Mostafa, Y.S., Psaltis, D., "Optical Neural Computers", *Scientific American*, Vol.256(3), March 1987, pp. 88-95.
- [Aho74] Aho, A.V., Hopcroft, J.E., Ullman, J.D., "Chapter 10: NP-Complete Problems", *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, Don Mills Ontario, 1974.
- [Aho77] Aho, A.V., Garey, M.R., Hwang, F.K., "Rectilinear Steiner Trees: Efficient Special-Case Algorithms", *Networks*, Vol.7, 1977, pp. 37-58.
- [Aker72] Akers, S., "Chapter 6: Routing", *Design Automation of Digital Systems: Theory and Techniques*, Vol.1, ed. Breuer, M.A., Prentice Hall, Englewood Cliffs New Jersey, 1972.
- [Alle83] Allen, E.M., "YAPS: Yet Another Production System", *University of Maryland CS TR-1146*, College Park Maryland, December 1983.
- [ART86] *Automated Reasoning Tool Reference Manual*, Inference Corporation, Los Angeles California, 1986.
- [Aven83] Avenier, J.P., "Digitizing, Layout, Rule Checking--The Everyday Tasks of Chip Designer", *Proceedings of the IEEE*, Vol.71(1), January 1983, pp. 49-56.
- [Baas78] Baase, S., "Chapter 7: "Hard" (NP-Complete) Problems and Approximation Algorithms", *Computer Algorithms*:

- Introduction to Design and Analysis*, Addison-Wesley Publishing Company, Don Mills Ontario, 1978.
- [Benn82] Bennett, J., Buchannan, B., Cohen, P., Fisher, F., "Chapter 7: Applications-Oriented AI Research: Science", eds. Barr, A., Feigenbaum, E.A., *The Handbook of Artificial Intelligence*, Vol.2, William Kaufman, Inc., Los Altos California, 1982.
- [Brat86] Bratko, I., *Prolog Programming for Artificial Intelligence*, International Computer Science Series, Addison-Wesley Publishing Company, Don Mills Ontario, 1986.
- [Breu83] Breuer, M.A., Carter, H.W., "Chapter 15: VLSI Routing", ed. Rabbat, G., *Hardware and Software Concepts in VLSI*, Von Nostrand Reinhold Company, Inc., New York New York, 1983.
- [Burs83]. Burstein, M., Pelavin, R., "Hierarchical Wire Routing", *IEEE Transactions on Computer-Aided Design*, Vol.CAD-2(4), October 1983, pp. 223-234.
- [Cloc81] Clocksin, W.F., Mellish, C.S., *Programming in Prolog*, Springer-Verlag, New York New York, 1981.
- [Deas86] Deas, R.D., *An Idiomatic Framework for the Automated Synthesis of Topographical Information From Behavioural Specifications*, University of Edinburgh Ph.D. Thesis, October 1986.
- [Deut76] Deutsch, D.N., "A 'Dogleg' Channel Router", *ACM IEEE 13th Design Automation Conference, 1976*, pp. 425-433.

- [Fike85] Fikes, R., Kehler, T., "The Role of Frame-Based Representation in Reasoning", *Communications of the ACM*, Vol.28(9), September 1985.
- [Forg79] Forgy, C.L., *OPS4 User's Manual*, Carnegie Mellon Univeristy CMU-CS-79-132, Pittsburg Pennsylvania, July 1979.
- [Forg81] Forgy, C.L., *OPS5 User's Manual*, Carnegie Mellon Univeristy CMU-CS-79-135, Pittsburg Pennsylvania, July 1981.
- [Gare79] Garey, M.R., Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco California, 1979.
- [Hama84] Hamachi, G.T., Ousterhout, J.K., "A Switchbox Router with Obstacle Avoidance", *ACM IEEE 21st Design Automation Conference*, 1984, pp. 173-179.
- [Hana66] Hanan, M., "On Steiner's Problem with Rectilinear Distance", *J. SIAM Applied Mathematics*, Vol.14(2), March 1966, pp. 255-265.
- [Haye83] Hayes-Roth, F., Waterman, D., Lenat, D., *Building Expert Systems*, Addison-Wesley Publishing Company, Don Mills Ontario, 1983.
- [High69] Hightower, D.W., "Solution to the Line Routing Problems in the Continuous Plane", *ACM IEEE 6th Design Automation Workshop*, 1969, pp. 1-24.
- [High80] Hightower, D.W., Boyd, R., R.L., "A Generalized Channel



- Router", *ACM IEEE 17th Design Automation Conference*, 1980, pp. 12-21.
- [Hong83] Hong, S.J., Nair, R., "Wire-Routing Machines--New Tools for VLSI Physical Design", *Proceedings of the IEEE*, Vol.71(1), January 1983, pp. 57-65.
- [Jenn84] Jennings, P.I., Hurst, S.L., McDonald, A., "Highly Routable ULM Gate Array and Its Automated Customization", *IEEE Transactions on Computer-Aided Design*, Vol.CAD-3(1), January 1984, pp. 27-39.
- [Joob85] Joobbani, R., Siewiorek, D.P., "WEAVER: A Knowledge-Based Routing Expert", *ACM IEEE 22nd Design Automation Conference*, 1985, pp. 266-272.
- [Joob86] Joobbani, R., *An Artificial Intelligence Approach to VLSI Routing*, Kluwer Academic Publishers, Hingham Massachusetts, 1986.
- [Keef86] Keefe, M.M., Kendall, J., "An Expert System for Routing in VLSI", *Canadian Conference on Very Large Scale Integration*, October 1986, pp. 337-342.
- [Kern73] Kernighan, B.W., Schweikert, D.G., Persky, G., "An Optimum Channel Routing Algorithm for Polycell Layouts of Integrated Circuits", *ACM IEEE 10th Design Automation Workshop*, 1973, pp. 50-59.
- [Kirk83] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., "Optimization by Simulated Annealing", *Science*, Vol.220(4), May 1983, pp. 671-679.

- [Lee61] Lee, C.Y., An Algorithm for Path Connection and its Applications, *IRE Transactions on Electronic Computers*, Vol.EC-10, 1961, pp. 346-365.
- [Leis81] Leiserson, C.E., Pinter, R.Y., "Optimal Placement for River Routing", *VLSI Systems and Computations*, eds. Kung, H.T., Sproull, B., Steel, G., 1981, pp. 127-151.
- [Mare84] Marek-Sadowska, M., "An Unconstrained Topological Via Minimization Problem for Two-Layer Routing", *IEEE Transactions on Computer-Aided Design*, Vol.CAD-3(3), July 1984, pp. 184-190.
- [Mare85] Marek-Sadowska, M., "Two-Dimensional Router for Double Layer Layout", *ACM IEEE 22nd Design Automation Conference*, 1985, pp. 117-123.
- [McDe81] McDermott, J., "R1: The Formative Years", *AI Magazine*, Vol.2(2), 1981, pp. 21-29.
- [Mead80] Mead, C., Conway, L., *Introduction to VLSI Systems*, Addison-Wesley Publishing Company, Inc., Don Mills Ontario, 1980.
- [Mukh86] Mukherjee, A., "Chapter 9: VLSI Design Tools", *Introduction to NMOS and CMOS VLSI Systems Design*, Prentice Hall Inc., Toronto Ontario, 1986.
- [Oust84] Ousterhout, J.K., Hamachi, G.T., Mayo, R.N., Scott, W.S., Taylor, G.S., "Magic: A VLSI Layout System", *ACM IEEE 21st Design Automation Conference*, 1984, pp. 152-159.
- [Part86] Partridge, D., "The Scope and Limitations of Expert Systems

Technology”,

*6th International Workshop on Expert Systems and Applications*, Agence de l'Information, Paris France, 1986, pp. 1543-1553.

- [Pers78] Persky, G., Deustch, D.N., Schweikert, D.G., “LTX- A Minicomputer-Based System for Automatic LSI Layout”, *Journal of Design Automation and Fault-Tolerant Computing*, May 1978; pp. 217-255.
- [Pint81] Pinter, R.Y., “Optimal Routing in Rectilinear Channels”, *VLSI Systems and Computations*, eds. Kung, H.T., Sproull, B., Steel, G., 1981, pp. 160-177.
- [Rive81] Rivest, R.L., Baratz, A., Miller, G., “Provably Good Channel Routing Algorithms”, *VLSI Systems and Computations*, eds. Kung, H.T., Sproull, B., Steel, G., 1981, pp. 160-177.
- [Rive82] Rivest, R.L., Fiduccia, C.M., “A 'Greedy' Channel Router”, *ACM IEEE 19th Design Automation Workshop*, 1982, pp. 418-424.
- [Rubi74] Rubin, F., “The Lee Connection Algorithm”, *IEEE Transactions on Computers*, Vol.C-23, 1974, pp. 907-914.
- [Sche86] Schediwy, R.R., *A CMOS Cell Architecture and Library*, University of Calgary Computer Science MSc. Thesis, December 1986.
- [Sech85] Sechen, C., Sangiiovanni-Vincentelli, A., “The TimberWolf Placement and Routing Package”, *IEEE Journal of Solid-State*

*Circuits*, Vol.S-20(2), April 1985, pp. 510-522.

- [Smit84] Smith, D.C., Noto, R., Borgini, F., Sharma, S.S., Werbickas, J.C., "The Variable Geometry Automated Universal Array Layout System (VGAUA)", *IEEE Transactions on Computer-Aided Design*, Vol.CAD-3(1), January 1984, pp. 20-26.
- [Souk79] Soukup, J., "Global Router", *ACM IEEE 16th Design Automation Conference*, 1979, pp. 481-484.
- [Souk81] Soukup, J., "Circuit Layout", *Proceedings of the IEEE*, Vol.69(4), October 1981, pp. 1281-1303.
- [Souk86] Soukup, J., Private Discussion, 1986.
- [Stei84] Steinberg, L.I., Mitchell, T.M., "Knowledge Based Approach to VLSI CAD: The Redesign System", *ACM IEEE 21st Design Automation Conference*, 1984, pp. 412-418.
- [Szym85] Szymanski, T.G., "Dog Leg Channel Routing is NP-Complete", *IEEE Transactions on Computer-Aided Design*, Vol.CAD-4(1), January 1985, pp. 31-40.
- [Ting83] Ting, B.S., Tien, B.N., "Routing Techniques for Gate Array", *IEEE Transactions on Computer-Aided Design*, Vol.CAD-2(4), 1983, pp. 301-312.
- [Ullm84] Ullman, J.D., "Computational Aspects of VLSI", *Computer Science Press*, Rockville Maryland, 1984.
- [VanC76] Van Cleemput, W.M., "On the Topological Aspects of the Circuit Layout Problem", *ACM IEEE 13th Design Automation Conference*, 1976, pp. 441-450.

- [Vecc83] Vecchi, M.P., Kirkpatrick, S., "Global Wiring by Simulated Annealing", *IEEE Transactions on Computer-Aided Design*, Vol.CAD-2(4), October 1983, pp. 215-222.
- [Yosh82] Yoshimura, T., Kuh, E.S., "Efficient Algorithms for Channel Routing", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.CAD-1(1), January 1982, pp. 25-35.