

AN ANALYSIS OF THE SINGLE ROW TRANSFORMATION  
(SRT) APPROACH TO VLSI CHANNEL ROUTING

CENTRE FOR NEWFOUNDLAND STUDIES

**TOTAL OF 10 PAGES ONLY  
MAY BE XEROXED**

(Without Author's Permission)

KUMAR VENGUSWAMY, B.E.









AN ANALYSIS OF THE  
SINGLE ROW TRANSFORMATION (SRT)  
APPROACH TO  
VLSI CHANNEL ROUTING



BY

Kumar Venguswamy. B.E.

A thesis  
submitted to the School of Graduate Studies  
in partial fulfillment of the requirements for  
the degree of Master of Science

Department of Computer Science  
Memorial University of Newfoundland  
St. John's, Newfoundland, Canada, A1C 5S7

July 1990



The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-65293-4

## Abstract

This thesis addresses the problem of detailed routing of Very Large Scale Integrated (VLSI) circuits using channel routing. Recently, a novel approach to detailed VLSI routing has been proposed by Wong and Kwok [Wong88]. This method, called the *Single Row Transformation* (SRT) approach, requires three steps, namely, Forward Transformation (FT), Single Row Routing (SRR) and finally Backward Transformation (BT). When this apparently simple approach was looked at more closely, it was realised that this approach has many hidden problems, the most important being those posed by the crossovers in the SRR layout while carrying out the BT step. Thus, although SRT based routing appeared to be a potential approach, further research was required. The goal of this thesis is to make a study and analysis of the SRT approach as it applies to channel routing, and to use the results of the analysis to design a channel router.

Although different FT-BT pairs are possible, only straightforward pairs are practical, and even for simple FT-BT pairs the problems posed by crossovers in the SRR solution necessitate a minimum crossover routing, calling for the design of new algorithms, because most of the existing SRR algorithms have been designed with the goal of track optimisation, not crossover minimisation. To facilitate the design of these algorithms, a taxonomy of SRR problems is useful. A proposed taxonomy classifies SRR problems into bipartite and non-bipartite problems, and further into *permutation* and *mixed* SRR problems. Based on this taxonomy, various algorithms for the different classes of SRR problems have been developed. Since some crossovers may be inevitable, effective crossover handling techniques are necessary. To this end different crossover handling techniques are discussed and a generalised crossover handling technique is proposed. The application of the analysis to the software design of a channel router and implementation of this software is also addressed.

## Acknowledgements

I am quite grateful to the Department of Computer Science and to the School of Graduate Studies of Memorial University of Newfoundland, for admitting me into the graduate program in Computer Science and for providing financial support in the form of a fellowship and teaching assistantships. I am also thankful to the Faculty of Engineering and Applied Sciences and to the Department of Physics whose employments were of great financial help.

I express my sincere gratitude to my advisor, Dr. Paul Gillard, for his supervision, helpful discussions and encouragement which have greatly helped me to complete this thesis. My sincere appreciation and thanks to Mrs. Jane Foltz, Head of the Department of Computer Science, for all the help generously extended and the advice offered throughout my graduate study.

My special thanks and appreciation to Dr. Wlodek Zuberek for his valuable association and enlightening discussions during the early stages of this thesis work. Helpful discussions with Dr. K. Vidyasankar were also appreciated.

The support from the systems groups was of great help. The prompt service from the general office was a delight and the help of all the secretaries was gratefully acknowledged. Finally, my thanks to my family and all my friends for their moral support and encouragement.

*To the Memory of My Father*

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	VLSI design automation . . . . .	1
1.2	A quick introduction to VLSI routing . . . . .	3
1.3	An outline of this thesis work . . . . .	5
<b>2</b>	<b>An Overview of Detailed Routing</b>	<b>8</b>
2.1	Detailed routing techniques . . . . .	9
2.1.1	Early routers . . . . .	9
2.1.2	A brief overview of Channel routing . . . . .	10
2.1.3	Channel routing models . . . . .	13
2.1.4	An overview of existing Channel routing algorithms . . . . .	15
2.2	Via minimisation in channels . . . . .	18
<b>3</b>	<b>An introduction to Single Row Routing</b>	<b>23</b>
3.1	An Introduction to the Single Row Routing Problem (SRRP) . . . . .	24
3.1.1	Origins and developments of the SRRP . . . . .	27
3.2	A taxonomy of algorithms for the SRRP . . . . .	30
3.2.1	Heuristic algorithms for the SRRP . . . . .	31

3.2.2	Enumeration algorithms for the SRRP . . . . .	38
3.2.3	Concluding remarks . . . . .	43
<b>4</b>	<b>The Single Row Transformation (SRT) Approach</b>	<b>45</b>
4.1	Different FT-BT pairs . . . . .	47
4.2	Some remarks on Wong's paper . . . . .	51
4.3	Problems posed by crossovers . . . . .	52
4.4	Crossover management techniques . . . . .	55
4.5	Pseudo stretches in SRR . . . . .	57
4.6	A Taxonomy of SRR problems . . . . .	59
<b>5</b>	<b>Bipartite Single Row Routing</b>	<b>62</b>
5.1	An overview of crossover free routing . . . . .	64
5.1.1	Formulations of the crossover free routing problem . . . . .	65
5.2	A New approach to the via free routing of Permutation Channels . . . . .	68
5.2.1	Permutation Channel / Permutation SRR problems . . . . .	69
5.2.2	Previous work on via free routing of a PCR <sup>2</sup> P . . . . .	70
5.2.3	Some new results on via free routing of a PCR <sup>2</sup> P . . . . .	73
5.2.4	Description of the new algorithm . . . . .	76
5.2.5	Extensions to the proposed algorithm . . . . .	79
5.3	MBSRR routing . . . . .	81
5.3.1	MBSRR routing using box procedure : extension . . . . .	82
5.3.2	MBSRR routing using interlocked set approach . . . . .	84
5.4	Modified Tarng's algorithm . . . . .	86
5.4.1	Tarng's algorithm revisited . . . . .	87

5.4.2	Details of the new algorithm TARNG-MOD . . . . .	90
<b>6</b>	<b>Non-Bipartite Single Row Routing</b>	<b>93</b>
6.1	Minimum crossover routing . . . . .	94
6.2	Topological via minimisation of channels . . . . .	98
6.2.1	A new heuristic for minimum node deletion . . . . .	101
6.3	Routing of PNSRR problems . . . . .	103
6.3.1	PNSRR with SNI+RNR approach . . . . .	105
6.3.2	Extension of Box-procedure to PNSRR . . . . .	109
6.3.3	Extension of TARNG-MOD to PNSRR . . . . .	110
6.3.4	PNSRR with straight nets (PNYSRR) . . . . .	115
6.3.5	Maximum conflicting overlap degree heuristic for the minimum node deletion problem . . . . .	119
6.3.6	Logical routing of PNSRR . . . . .	123
6.4	Routing of MNSRR problems . . . . .	127
6.4.1	MNSRR by extending previous approaches . . . . .	127
6.4.2	Routing of MNSRR by grouping into PNSRR . . . . .	128
<b>7</b>	<b>Backward Transformation</b>	<b>131</b>
7.1	Crossover Handling techniques . . . . .	132
7.1.1	Crossover collapsing . . . . .	140
7.2	Grid opening: A generalised crossover handling technique . . . . .	141
7.3	Fold back step of the BT . . . . .	145
<b>8</b>	<b>Software development of an SRT based router</b>	<b>148</b>
8.1	Structure of the software . . . . .	148



8.1.1	General router . . . . .	149
8.1.2	Special routers . . . . .	149
8.2	Implementation details and results . . . . .	151
<b>9</b>	<b>Conclusions</b>	<b>157</b>

# List of Figures

2.1	Example of a channel and a switchbox problem . . . . .	11
2.2	Horizontal and vertical constraint graphs of a channel . . . . .	12
2.3	Different routing models . . . . .	14
2.4	Constrained via minimisation of a 2-layer channel . . . . .	20
2.5	Unconstrained via minimisation of a 2-layer channel . . . . .	21
3.1	An example of an SRR problem . . . . .	25
3.2	Examples of some legal and illegal realisations of an SRRP. . . . .	26
3.3	Examples of crossovers of different types. . . . .	27
3.4	An interval representation and physical realisation of an SRRP. . . . .	29
3.5	Routing a SRR problem with Tarng's algorithm. . . . .	34
3.6	Concept of <i>groups</i> . . . . .	35
3.7	A 2-group SRR problem routed with Tarng's algorithm . . . . .	36
3.8	A 2-group SRR problem routed with Du's algorithm. . . . .	37
3.9	Interval graph representation and groups of an SRR problem . . . . .	39
3.10	Feasible orders generation for a B node. . . . .	42
4.1	Channel routing by the SRT approach . . . . .	46
4.2	Different FT-BT pairs for the SRT based routing . . . . .	48

4.3	Steps of the <i>abcd</i> type SRT . . . . .	49
4.4	An example of an easily alignable crossover . . . . .	54
4.5	A crossover which needs a careful alignment . . . . .	56
4.6	Translation of channel nets to SRR nets . . . . .	58
4.7	A taxonomy of SRR problems . . . . .	61
5.1	Some examples of SRRP nets. . . . .	64
5.2	Some examples of Interlocking. . . . .	66
5.3	A Permutation Channel Routing Problem (PCRP) . . . . .	69
5.4	Permutation Channel routing by SRT approach. . . . .	71
5.5	Minimum number of <i>ascending chains</i> approach . . . . .	73
5.6	Properties of the nets of a PCRP . . . . .	75
5.7	Example of MBSRR routing . . . . .	83
5.8	Legal structure of nets in the same street of a bipartite SRR . . . .	84
5.9	Examples of single and multi-interlocked MBSRR . . . . .	85
5.10	Tarng's algorithm's failure in crossover free routing. . . . .	88
5.11	TARNG-MOD algorithm based routing of a MBSRR . . . . .	92
6.1	Realisation with different crossovers leading to same via count . . . .	95
6.2	Topological via minimisation of a channel . . . . .	102
6.3	Node deletion of a PNSRR using MOD heuristic . . . . .	104
6.4	Node deletion of a MNSRR using MOD heuristic . . . . .	105
6.5	Routing of a PNNSRR using SNI+RNR approach . . . . .	106
6.6	Placement of a residual net . . . . .	108
6.7	Routing a PNNSRR problem by box procedure . . . . .	111

6.8	PNNSRR problem with a RRR crossing routed by box procedure . . . . .	112
6.9	Tarnq's order based level assignment . . . . .	114
6.10	Routing a PNNSRR problem by TARNG-MOD . . . . .	116
6.11	Routing a PNYSRR problem . . . . .	118
6.12	A PNYSRR problem with multiple crossovers on the straight net . . . . .	120
6.13	An example where MOD heuristic fails . . . . .	121
6.14	Logical routing of a PNYSRR problem . . . . .	125
6.15	MNSRR routing by grouping . . . . .	130
7.1	Relation between crossover and adjacent nets - Class A . . . . .	134
7.2	An example of aligning an easy crossover . . . . .	135
7.3	Relation between crossover and adjacent nets - Class B . . . . .	137
7.4	An example of aligning an easy 2-bound crossover . . . . .	138
7.5	Handling of a difficult crossover by split-align technique . . . . .	139
7.6	An example of split-align that needs an external dogleg . . . . .	140
7.7	Handling a crossover on a straight net . . . . .	142
7.8	Grid opening technique to handle crossovers . . . . .	144
7.9	Different ways of folding back an SRR layout . . . . .	146
8.1	Software structure of a general SRT router . . . . .	150
8.2	Software structure for special channel routers . . . . .	152
8.3	A channel problem routed by the implemented router . . . . .	155
8.4	A channel problem routed by the implemented router . . . . .	156

# Chapter 1

## Introduction

### 1.1 VLSI design automation

The highly complex electronic systems of today are usually built using a number of Very Large Scale Integrated (VLSI) chips; the design of such VLSI chips is a time-consuming and complex task. In fact, the development time for an electronic product containing VLSI components may be longer than the projected lifetime of the product and the investment in the design process may not be recovered before the product becomes obsolete. Given this situation, cost effective VLSI designs must be produced as quickly as possible, be functionally correct in the first pass and meet specifications without lengthy tuning and design iterations. One way to achieve these goals is to automate the VLSI design process.

Sophisticated Computer Aided Design (CAD) tools are necessary to cope with this complex task of VLSI design automation. As improved circuit fabrication technologies make higher levels of integration possible, CAD tools to support the design and testing of these complex chips become mandatory. In fact, VLSI design automation has steadily accelerated to keep pace with the innovations in

architectures and advances in circuit fabrication technologies in the past decade. Today the scope of VLSI Design Tools is very broad and encompasses much of the design process from design specification, logic design, partitioning, placement, routing, simulation, and so on until the final mask is realised.

Because of the complexity involved, VLSI design is usually attempted in a hierarchical fashion and many of the modern CAD tools are based on and fully exploit such hierarchical schemes. Hierarchical decomposition reduces the complexity by breaking the overall circuit into a number of smaller subcircuits such that automatic design is computationally feasible. Top-down circuit partitioning and bottom-up circuit layout is the traditional approach. The top-down decomposition stops when a subcircuit can be effectively laid out by the designers using interactive graphics editors or until the layout can be automatically generated by programs (so-called module generators or cell compilers). The bottom-up layout process then takes the set of layouts for these bottom subcircuits and places and interconnects them hierarchically, until the complete circuit is laid out.

A large, often dominant part of the cost and time required to design a complex chip is consumed in the *physical design* phase and in particular in the *layout* step which is the task of placing a set of modules (*placement*) on a chip and realising a set of interconnections between these modules (*routing*). Clearly, placement and routing are closely related to each other and a poor placement could make the subsequent routing difficult or impossible. The ideal way of solving the layout problem as a whole is to develop an algorithm that simultaneously and globally considers the position of the modules and all the interactions between the required interconnections. However, because of the tremendous complexity involved, the layout problem is usually solved using the two sequential steps of placement and routing. For the purposes of this thesis, the interest and the focus is solely on the routing step.

## 1.2 A quick introduction to VLSI routing

Automated routing had its origins in the design of Printed Circuit Boards (PCBs) which in the 70's had typically hundreds of nets and limited rules for interconnection. As electronic design methods improved, the early automated routing algorithms were adapted and numerous new algorithms were developed to handle multilayer PCB, LSI and VLSI routing. Furthermore, the routing techniques were influenced by the choice of technology (for example, the number of layers available for routing) as well as by the particular design style — full custom or semi-custom using gate-array, standard cell or general cell methodologies. The cell based design approach is popular and the entire chip can be considered as a tree made up of blocks of decreasing complexity from the root to the leaf nodes.

Routing or interconnection is one of the important steps in the physical design of VLSI circuits. One can visualise an Integrated Circuit (IC) chip as made up of a number of building blocks which may be macro cells, standard cells, gate arrays, functional blocks, *etc.*, depending upon the design style. These building blocks communicate with each other via pins located on their boundaries. Thus, the routing problem for an IC chip is mainly concerned with making the interconnection between the modules. Apart from the modules and the signal nets which interconnect these modules, a chip will have power and ground wiring which runs in a bus structure and feeds each of the blocks. Further, there is a padframe which surrounds the modules and provides pads for connections to the external pins on the chip. In general, most of the chip area not occupied by the modules is available for interconnecting the modules and is in general, referred to as the routing area or routing domain.

In an integrated circuit chip, there may be many such blocks and each block may have many pins on its boundaries. Thus there are many pins scattered

on the surface of the chip which belong to distinct sets, where each set of pins represents a *net*. The purpose of routing then is to connect electrically the pins which make up each net such that all nets are electrically isolated from each other. Each net will be made up of a number of wiring segments which will run on one of the layers available for routing, subject to some restrictions as dictated by the routing models. When the adjacent segments of a net are routed in different layers, *vias* are used at the points where the layer changeover occurs so that the entire net is electrically connected.

Because routing is a complex process in itself, it is usually attempted as a 2-step process, namely *global routing* and *detailed routing*. Global routing is a preliminary planning stage for the subsequent detailed routing. In the global routing phase, the overall complex shaped routing area (the wiring bay) is broken up into many smaller areas of simple and regular shape (say rectangles) and nets which will use these areas for routing are identified. In general, there may be more than one way to split the total routing problem into a set of disjoint detailed routing problems. Thus, at the end of global routing (also called loose routing) we have subproblems each of which is characterised by a set of pins, an interconnection pattern and an area for routing them using a chosen routing model. Note that the routing paths of the nets inside the routing areas is not determined yet. In the subsequent phase of detailed routing, the actual wiring of each of these small areas is realised such that all the building blocks on the chip get interconnected.

Based on the shape of the routing region and the location and nature of the terminals, the detailed routing problem has been studied using some popular models. If the routing area is a rectangle (no obstacles inside) with pins located on two parallel sides it is called the channel routing problem. The term *channel* usually refers to a rectangular straight channel, although T, X or L shaped channels are also possible. If terminals are allowed on all four sides of the rectangular routing region, then it is called the switchbox routing problem. Another model for detailed



routing which is popular in PCB design is the Single Row Routing Problem (SRRP) where all the pins to be connected line up along a single row and the routing area is on both sides of the line of nodes.

### 1.3 An outline of this thesis work

This thesis addresses the problem of detailed routing of VLSI chips. Channel routing is the most commonly used method for the detailed routing phase in many of the automated VLSI design systems and various channel routing algorithms exist for this purpose. Recently, a novel approach to detailed VLSI routing was proposed by Wong and Kwok [Wong88] which is applicable for routing channels, switchboxes or any polygon shaped routing region. This method, called the *Single Row Transformation* (SRT) approach, requires 3 steps. The first step, called the *Forward Transformation* (FT), converts the input detailed routing problem (say a channel) into an equivalent single row routing problem. In the second step, the Single Row Routing solver, the SRRP is solved and an SRR wiring layout is obtained. In the last step, called the *Backward Transformation* (BT), the SRR layout is folded back into a channel layout to obtain a realisation for the original channel problem. The main reason for using the 3-step approach instead of the straightforward single step approach, as explained in [Wong88], is to exploit the efficient routing algorithms available for SRRP. Wong *et al.* [Wong88] claim that the quality of the solution is comparable to that produced by the direct approach.

When this apparently simple approach was looked at more closely by working out different examples, it was realised that this approach has many hidden problems, the most important being the problems posed by crossovers in the SRR layout while carrying out the BT step. The reason for these problems is the fact that this approach attempts to make a bridge between the channel routing problem

and the Single Row Routing problem which so far has been studied as different problems with different underlying routing models. To the best of the author's knowledge, [Wong88] is the only work reported on SRT based routers so far. Thus, the goal of this thesis is to make a detailed study and analysis of the SRT approach as it applies to channel routing, and to use the results of the analysis to design a channel router based on the SRT approach.

The thesis is organised to first present the background material on channel routing and single row routing, and then to discuss the SRT approach, its problems and some solutions. Chapter 2 gives a brief review of the detailed routing techniques, especially channel routing, highlighting various routing models and algorithms for solving the channel routing problem. In Chapter 3 the single row routing problem is introduced and a taxonomy of the various algorithms that have been reported so far is presented. The inadequacy of the existing algorithms for the application in SRT based routers is also brought out. With this background, Chapter 4 introduces the SRT approach to VLSI routing. The FT and BT steps are discussed in more detail by considering different possible FT-BT pairs. Then, the hidden problems (especially the problems posed by crossovers in the SRR solution) in the apparently simple SRT approach are highlighted. Crossover management using the steps of crossover reduction and crossover handling is proposed. The concept of *pseudo stretches* in SRR nets is introduced and its applications indicated. Finally, a taxonomy of SRR Problems in the light of its application in SRT based routers is proposed. Chapter 5 considers the bipartite cases of SRR, namely, the Permutation Bipartite SRR and the Mixed Bipartite SRR. A new  $O(n)$  algorithm, called the *box procedure* [Veng90], for crossover free routing of a Permutation Bipartite SRR problem and hence the via free routing of the corresponding Permutation Channel, is proposed. For routing a Mixed Bipartite SRR problem, a modified and enhanced version of Tarn's algorithm [Tarn84], called TARNG-MOD is proposed. The approach uses 2-colouring of the Interval Overlap Graph of the SRR. Chapter

6 considers the non-bipartite cases of the SRR problem, namely, the Permutation Non-Bipartite SRR and the Mixed Non-Bipartite SRR problems. First a detailed analysis of the Topological via minimisation of channels is presented and a new heuristic approach to the minimum node deletion bipartite subgraph problem, is proposed. Then various algorithms are presented and compared to route the Permutation Non-Bipartite SRR problem with and without straight nets. In the case of the Permutation Non-Bipartite SRR the box procedure normally produces a minimum crossover solution. For routing the Mixed Non-Bipartite problem, a divide and conquer approach using the concept of *groups* [DuLi87] is proposed, where the Mixed Non-Bipartite SRR problem is broken into Permutation Non-Bipartite SRR problems and routed. Chapter 7 discusses the Backward Transformation step. Here different crossover handling techniques are discussed. A general crossover handling technique, called *grid open*, is described which is based on a pseudo elongated channel and local river routing. This technique is guaranteed to produce a final channel for any type and number of crossovers. Channel width reduction techniques are also discussed. Chapter 8 uses the analysis carried out in the previous chapters and presents the framework for the software design of a channel router based on the SRT approach. Implementation details and experimental results are also presented. Chapter 9 provides the conclusions and suggests directions for future work.

## Chapter 2

# An Overview of Detailed Routing

This chapter gives a broad overview of detailed VLSI routing with emphasis on channel routing. To start, the scope of detailed routing in the VLSI layout process is presented and historical detailed routing techniques are mentioned. Then terminology related to channel routing is presented and different channel routing models are discussed. Next an overview of different algorithms for channel routing is presented. Finally, the problem of via minimisation in channels is discussed.

The problem of VLSI circuit layout is usually partitioned into two sub-problems, namely, *placing* a set of modules and *routing* a set of interconnections between the modules. Placement algorithms try to place the modules in a manner that will minimise the circuit area and facilitate the routing phase. After placement, the routing phase consists of two parts, namely, *global routing* and *detailed routing*. detailed routing step takes the input from the global routing step and carries out the actual routing of nets within each channel thereby resulting in the physical layout.

## 2.1 Detailed routing techniques

The problem of routing or interconnection takes typically 30% of the total design time and 50% of the chip area and thus needs special attention in the CAD of VLSI circuits. The general VLSI routing problem as well as many of its subproblems and restricted versions are NP-complete [Szym85, Sarr87] problems.<sup>1</sup> Various routing models have evolved to formulate the constraints posed by the manufacturing technology and to ease the design of CAD tools. Thus, different routing algorithms use different underlying routing models. The first approaches to routing started with the maze and line search routers, but in the past decade channel routing has clearly established itself as the standard method for detailed routing of VLSI chips.

### 2.1.1 Early routers

The earliest routers are the so called *area routers*, and Lee's algorithm [Lee61] was the earliest and most general method for routing. It finds the shortest path between two pins by using a wave propagation and labelling scheme, if a path exists. The main drawback is the excessive time and storage requirements. The modified versions [Aker67, Rubi74] use different speedup techniques (better labelling schemes, directed wavefront propagation *etc.*). Next came the *line search* routers [High69] which are similar to Lee routers but do not consider the entire matrix of grid points on the routing grid but use horizontal and vertical line sweep techniques instead, thereby reducing time and storage requirements. However, both area and line routers route one net at a time and do not consider the netlist as a whole. Because of this lack of information concerning the interaction between different nets,

---

<sup>1</sup>NP-completeness means that as the size of the problem (a good measure of size for the routing problem would be the number of nets) increases, the time required to solve the problem increases exponentially. No polynomial time algorithm for any member of the NP-complete family has been reported yet. If a polynomial time algorithm exists to solve one member of the family, then there is a polynomial time algorithm to solve all other members of the family.

sometimes an already routed net blocks the path of a subsequent net. Moreover, different ordering of nets as well the ordering of pins within the net may lead to different solutions and these routers may fail even in very simple situations. These drawbacks led to the development of *channel routers*.

### 2.1.2 A brief overview of Channel routing

Channel routing is the workhorse of many of the present day CAD systems that support automated layout. The channel routing problem was formulated by Hashimoto and Stevens [Hash71] in 1971, who also gave the *Left Edge Algorithm*. Thereafter, different channel algorithms have been reported [Deut76, Rive82, Yosh82, Burs83, Joob86]. Channel routers are parallel routers; they consider the input netlist as a whole and thus the ordering of nets in the input is not relevant. Further, channel routers are smarter in the sense that they capture the interaction between the different nets using the vertical constraint graph (VCG), before proceeding with the detailed routing. Cycles in the VCG pose problems, however, which in some cases could be managed using doglegging. Generally, 100% routing is not guaranteed. Greedy channel routers making use of heuristic rules search for local optimality, consequently creating situations where decisions made earlier cause the channel to be unroutable or increase the width of the channel more than necessary. A recent trend is to use knowledge based expert systems [Joob86] for VLSI routing.

A *channel* normally refers to a rectangular shaped routing area with straight edges and terminals located on two of its sides and no obstacles inside the routing domain. While other complex shapes are possible, the rectangular straight channel is the fundamental shape. The word *channel* from now onwards will mean a rectangular straight channel, an example of which is shown in Figure 2.1. The switchbox is another important routing problem where pins are allowed on all four sides of the rectangle. The objective of the channel router is to interconnect the



intersection of the intervals. That is, if there is an edge in the HCG between nodes  $a$  and  $b$ , it means that the horizontal stretches of the nets  $N_a$  and  $N_b$  intersect.

Another important property of a channel is the *local density*. The local density at any column of a channel is equal to the number of nets which cut a vertical line drawn at that column. The overall *density* of a channel is the maximum of its local densities. The density is the lower bound on the number of horizontal tracks needed (also called channel width) in any realisation of the channel. The *span* of a net refers to the horizontal stretch of a net in the channel. For a straight net which connects nodes in the same column the span is zero. A *left net* connects a pin on the top side to a pin to its left on the bottom side. A *right net* is defined similarly. All pins of a *local net* lie on the same side of the channel. A permutation net has only 2 pins which are on the opposite sides of the channel. Thus, a permutation channel can have only straight, left and right nets, whereas a mixed channel has one or more local nets in addition to the permutation nets.

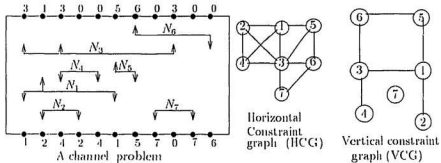


Figure 2.2: Horizontal and vertical constraint graphs of a channel

The VCG captures the ordering information between the nets. The nodes of the VCG represent the nets, and the edges represent the precedence constraints. Thus any cycle in the VCG means there is an unresolvable conflict in the ordering and all the constraints cannot be satisfied without breaking the loop. An important



feature of the VCG is the length of the longest path, which poses a limit on the minimum number of horizontal tracks needed. In fact, for simple routing models the channel width  $\geq \max\{\text{channel density, length of longest path of the VCG}\}$ .

### 2.1.3 Channel routing models

Routing models can be classified based on the number of layers available for routing the wire segments and the restriction on their course of run. The most common are 2-layer schemes, while three or more layer schemes are becoming important with the advances in VLSI technology. The classification here assumes a 2-layer model, where 2 layers, called metal 1 and metal 2, are available for routing. Another point to note is that in some technologies the pins are accessible from all the routing layers whereas in others the pins are accessible from only one routing layer, requiring that the first and the last routing segments for each net lie in the same layer. The rest of the segments could run in other layers using vias wherever needed.

The *HV model* is a reserved layer model in which all the segments in one layer must run in the horizontal direction while all the segments in the other layer must run in the vertical direction. The HV-model simplifies the routing job and guarantees a solution if there are no loops in the VCG but introduces many vias. The next model is the *2-way model* which allows both horizontal and vertical runs in both layers. This reduces the number of vias significantly. A further relaxation is the *overlap model* which allows parallel runs (run along the same horizontal or vertical grid) of two or more nets in different layers. The advantage of overlap is that it could result in a decrease in channel width, ( $w \geq d/2$ ), where  $d$  is the channel density. The disadvantage is that long runs of overlap could result in crosstalk due to capacitive coupling. Hambrusch *et al.* [Ham85] describe in detail the lower bounds for channel with overlap models for multilayer cases. Another is the *knock knee* model, where two wiring segments (of different layers) are allowed

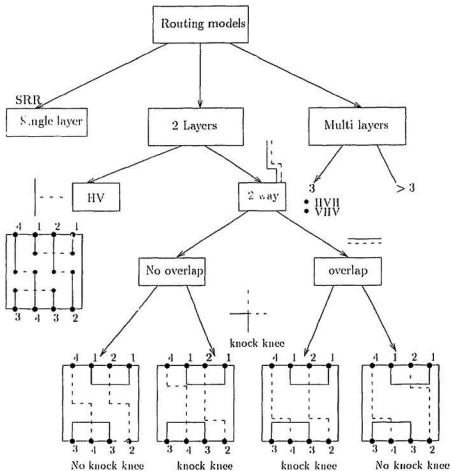


Figure 2.3: Different routing models

to share a grid point. Algorithms and simplified proofs for channel routing in knock-knee mode have been reported [Mehl86]. In some models both overlap (grid line sharing by two nets) and knock-knee (grid point sharing by two nets) are allowed. Rivest *et al.* [Rive81] present provably good algorithms for this model. Figure 2.3 depicts realisations of a channel problem using these different models. The solutions differ in terms of the channel width, number of vias and the total wire length. One important thing to keep in mind when comparing the quality of routing of the solutions produced by different routing algorithms is the fact that the underlying routing model in each of these algorithms may be different and it is not proper to compare directly the results of algorithms which use quite different models, although such sweeping comparisons are made, for example in [Wong88].

#### 2.1.4 An overview of existing Channel routing algorithms

From the earliest *left edge* algorithm of Hashimoto and Stevens [Hash71] various improved and efficient algorithms [Deut76, Rive82, Yosh82, Burs83, Joob86] have been proposed for channel routing. Some of these algorithms have been extended to complex shaped channels, channels with movable terminals, switchbox problems, multilayer routing, *etc.* The left edge channel router attempts to maximise the placement of horizontal segments in each track. The router uses a left edge sorted order for the nets and hence its name. It uses the HIV model and always yields optimum channel width if there are no vertical constraints. However, the presence of vertical constraints often yields sub-optimal results. Loops in the VCG result in non-routability of the channel and this necessitates splitting of nets called *doglegging*.

In the *restrictive* channel routing problem, the entire horizontal segment of a net has to lie at the same level. The idea of the *dogleg* router was proposed by Deutsch [Deut76] which removes the heavy constraint placed by restrictive routing.

Here a net is split between different tracks, i.e., the net occupies more than one horizontal track and the vertical pieces of wire connecting the different horizontal segments are referred to as doglegs. Doglegs can be internal or external. An external dogleg lies in a column outside the span of the net and is needed if a net has *detours*. Introduction of doglegs usually results in a decrease in channel width, especially when there is long path in the VCG of the channel. Further, doglegs resolve the loops in the VCG and increase the routability of channels. However, each dogleg introduces two vias and this leads to increased area, resistance and reduced speed and reliability. Thus, it is desirable to limit the number of doglegs. Deutsch proposed a very simple and efficient way to introduce doglegs only at the columns where the net has a pin (except the left and right pin). Later, Deutsch improved his algorithm by introducing a controlling parameter called *range* which defines the minimum number of subnets that must be assigned on the current track. As the range parameter increases, fewer doglegs are introduced.

In recent years heuristics have been utilised in channel and switchbox routers. The *greedy* algorithm by Rivest [Rive82] was the first of these attempts to use a few (less than 10) rules to implement a channel router. There are several modified versions of the greedy algorithm. The greedy channel router scans the channel in a left-to-right, column by column manner, completing the wiring within a column before proceeding to the next. It may place a net on more than one track and have a vertical line crossing more than one horizontal segment of the same net. In each column, the router performs the following steps:

- Brings in the nets in the top and bottom of the channel, using the shortest vertical line, to either an empty row or a row that contains the net.
- Frees up as many tracks as possible by making vertical connecting jogs that collapse nets currently occupying more than one track.
- Reduces the distance between the tracks occupied by nets still occupying

more than one track.

- Moves a net up if its next pin is on the top of the channel or down if its next pin is on the bottom.
- Adds a new track if the channel is full and a pin could not enter the channel.

The router usually completes the routing, even in the presence of cyclical vertical constraints, often using no more than one extra track than the channel density (minimum number of tracks). However, the greedy algorithm suffers from some problems. Since it is greedy it searches for local optima, consequently creating situations where decisions made earlier may make the channel unroutable or increase the width of the channel.

Two algorithms proposed by Yoshimura and Kuh [YoKu82], attempt the placement of the nets on tracks in a different way. The first algorithm attempts to minimise the longest path in the VCG by combining those tracks that minimise the path through the VCG. This merging operation modifies the VCG by treating those nets which do not constrain each other as a single node. Each set of merged nodes can be assigned on the same track. The second algorithm achieves longest path minimisation through matching techniques on a bipartite graph. Both algorithms report better results than the dogleg router.

Hierarchical router [Burs83] is based on a *divide and conquer* approach and was the first router to automatically complete Deutsch's difficult channel example in 19 tracks. The general approach in hierarchical routing is to divide the routing area into subareas each of which is a  $2 \times 2$  grid. All terminals are located in the center of the basic cell; then either linear integer programming or dynamic programming is used to decide an optimal interconnection pattern between terminals of different cells. The linear equations are reduced such that the integer programming problems are fixed, independent of the number of nets to be wired.

There are 12, 12 and 4 patterns for connecting signal nets with two terminals, three terminals and four terminals respectively in  $2 \times 2$  rectangular cells. The division process proceeds until the single cell resolution is reached completing the routing. Refinement is performed on the routing of all nets at every level of the hierarchy.

A totally different approach was taken by Joobbani and Siewiorek [Joob86] in the development of a channel router called Weaver. Weaver combines algorithmic approaches like the vertical constraint graph with simple deductive and expert knowledge. Since there are many metrics that need optimisation, Weaver provides an *expert* on each: constraint propagation, wire length, congestion, *etc.* These experts observe the problem and make suggestions based on their own area of concern. For example, the via expert suggests alternatives to the routing of a completed net, and attempts to remove unnecessary vias by changing the layers of some wire segments. This expert not only reduces the number of vias, but also frees up some routing space on the reassigned segment's original layer. A scheduler, which is an expert system as well, then decides on the best application of all the suggestions received. In addition to its ten experts, weaver provides the user with the ability to act as an additional resource. The user can override any system decision and either pre-route or delete wire segments. Weaver is able to route Burstein's difficult switchbox automatically using fewer vias and less wiring than a manually guided solution obtained by the greedy router.

## 2.2 Via minimisation in channels

There are a number of reasons why the number of vias in a channel layout should be kept to a minimum. In integrated circuit processing, more vias usually lead to a lower yield. Further, every via has an associated resistance which affects the circuit performance. The size of a via is usually larger than the width of

the wires and hence more vias mean more routing space. There is an inverse correlation between the number of vias used by a router and the completion rate of the router. With these drawbacks in mind, via-minimisation algorithms have been studied extensively under the models of *Constrained Via Minimisation* (CVM) and *Unconstrained Via Minimisation* (UVM).

In constrained via minimisation, an initial layout is assumed to be available and the goal is to reduce the number of vias in the input layout by reassigning some nets and net segments to the opposite layer. Thus, CVM does not alter the layout of the input solution but attempts a selective layer reassignment. An example of CVM in a 2-layer channel is shown in Figure 2.1.a. Hashimoto and Stevens [Hash71] first formulated the CVM problem for the Printed Circuit Board design. Subsequently many algorithms have been proposed [Kaji80, Cies81] for CVM in a 2-layer environment. The two layer CVM problem is not NP-complete and can be solved in polynomial time. Chang and Du [ChDu88] formulated the three layer CVM problem and showed that it is an NP-complete problem and presented an heuristic algorithm as well.

The result obtained by the CVM approach is minimal only with respect to the given input layout and may not be the optimum solution for the problem. For example, an initial layout with 10 vias shown in Figure 2.4.a translates into a 5-via solution after via minimisation whereas, an initial layout with 12 vias shown in Figure 2.4.b translates into a 3-via solution. Thus, to consider the problem as a whole instead of acting on a discrete realisation the goal of topological via minimisation was proposed.

The *Topological Via Minimisation*, also called Unconstrained Via Minimisation, considers routing and via minimisation as an integrated step and was proposed by Hsu [Hsu83]. Here the routing of a channel is split into two steps, namely, *topological routing* and *geometric mapping*. The topological routing step

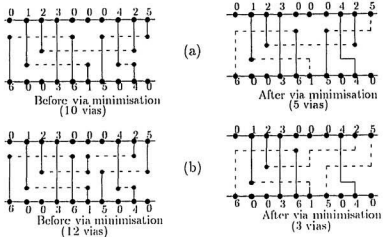
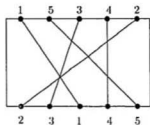


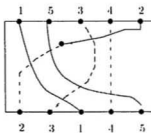
Figure 2.4: Constrained via minimisation of a 2-layer channel

attempts to find a minimum via solution for the channel, taking into account only the topological constraints and not the geometrical constraints (imposed by the technology), which are taken care of by the geometrical mapping step. Hsu captures the topological constraints between the nets in terms of the *circle graph*  $C_g$  of the channel. Given a channel, its circle graph can be obtained as follows. First obtain the *circular representation* of the channel, by traversing the nodes of the channel in a circular fashion. Using this representation, every 2-pin net of the rectangular channel would translate into a chord in the circular representation. If two nets cross in the channel, then the corresponding chords will intersect. The *circle graph* is drawn such that its nodes represent the chords (nets) and the edges capture the intersection of chords (crossing of channel nets), as illustrated in Figure 2.5. If the circle graph is bipartite then the channel can be routed without vias. Otherwise, only its maximum bipartite subgraph can be routed without vias and the remaining nets need vias. Absolute minimisation of vias may necessitate many detours in the channel. Topological via minimisation of 2-layer, 2-pin nets has been

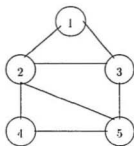




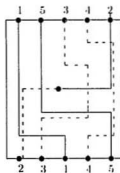
A Channel Problem



A Topological  
Routing with one via



Circle Graph  
 $C_g$  of the channel.



Final layout after  
geometric mapping

Figure 2.5: Unconstrained via minimisation of a 2-layer channel

shown to be NP-complete [Hsu83, Sadw84]. For the special case of permutation channels, polynomial algorithms have been proposed [Sarr89, Rim89]. Topological via minimisation will be discussed in more detail in Chapter 6.

## Chapter 3

# An introduction to Single Row Routing

In this chapter the Single Row Routing Problem (SRRP) is introduced. Although many algorithms for the SRRP have been proposed, no work has been reported yet on a comparative study and a comprehensive taxonomy of these different SRR algorithms. The focus of this chapter is a taxonomy of SRR algorithms that have been reported to date, together with a brief description of these algorithms. Further, the inadequacy of the existing SRR algorithms to achieve the *minimum crossover routing* demanded by the SRT environment is indicated and this calls for the design of new or modification of existing SRR algorithms. Particular attention is paid to and a detailed analysis is presented for Tarng's algorithm [Tarn84].

### 3.1 An Introduction to the Single Row Routing Problem (SRRP)

The Single Row Routing Problem (SRRP) is a very much restricted subset of the general routing problem. It contains a set  $V = \{1, 2, 3 \dots n\}$  of evenly spaced nodes which lie along a single row. The set  $L = \{N_1, N_2, \dots N_m\}$  is called the *netlist* and each net  $N_i$  ( $1 \leq i \leq m$ ) in  $L$  contains a set of two or more of the  $n$  nodes, which are to be connected together to make them electrically common. Each node belongs to exactly one net; i.e.,  $N_i \cap N_j = \Phi$  if  $i \neq j$ . For example, an SRRP with  $m = 4$  and  $n = 8$  ( $m$  = total number of nets and  $n$  = total number of nodes) can have the netlist (or interconnections)  $N_1 = (1, 4)$ ,  $N_2 = (2, 7)$ ,  $N_3 = (3, 6)$  and  $N_4 = (5, 8)$ . A *realisation* of the netlist  $L$  is to be made on a *single layer* by using non-overlapping wires that are composed of only vertical and horizontal segments. One possible realisation of the netlist given above is shown in Figure 3.1.

The row on which the nodes lie is called the *node axis* and the area above the node axis is called the *upper street* and the area below the node axis is called the *lower street*. The maximum number of horizontal tracks allowed in the upper (lower) street is called the upper (lower) *street capacity* while the actual number of tracks used in the upper (lower) street in any particular realisation is called the upper (lower) *street congestion*. The symbols  $C_{us}$  and  $C_{ls}$  stand for the upper and the lower street congestion respectively. For example, the realisation shown in Figure 3.1 has  $C_{us} = 1$  and  $C_{ls} = 2$ .

In addition to using only non-overlapping rectilinear segments, a realisation usually must also not have any left-right zigzagging of the nets. This means that a vertical cut made at any node can intersect at most one horizontal segment from each net. [Ragh83] refers to the left-right zigzagging as backward moves or backmoves. Only those realisations without backmoves are considered as legal in

this discussion. Some examples of legal and illegal realisations are shown in Figure 3.2. However, up and down zigzagging of the nets is allowed, *i.e.*, the nets can change over from one street to another in the inter-node space. Such transitions are called *crossovers*. Each net may have zero or more crossovers and zero or more nets can crossover at the same inter-node space. The maximum number of crossovers in any of the inter-node space is called the *crossover bound* usually denoted as  $K$ . Some realisations illustrating various types of crossovers are shown in Figure 3.3.

Given a netlist  $L$  containing  $m$  nets, there are  $m!$  possible realisations. Each realisation may have a different street congestion and a different number of crossovers. The SRRP has been studied mainly with the objective of minimising the tracks used. An optimum realisation is one which minimises the street congestion in both streets. Thus optimality with respect to tracks used (street congestion) aims at minimising the value of  $Q$  where  $Q = \max\{C_{us}, C_{ls}\}$ . Another possible optimality criterion could be the minimisation of the crossovers. Little work has been reported in this direction yet and we will return to this aspect in Chapter 4 and Chapter 6.

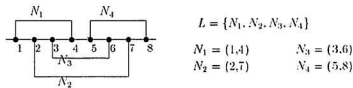


Figure 3.1: An example of an SRRP problem

A given routing problem (*e.g.* a channel or a switchbox problem) can always be converted into an equivalent SRRP. For example, a channel could be easily converted to an SRRP by unfolding the channel and placing the two longer sides, side by side. In a similar way, a switchbox can be unfolded into a single row of nodes. One of the advantages of the SRRP when compared to other routing

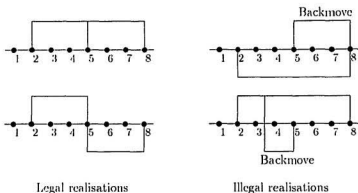


Figure 3.2: Examples of some legal and illegal realisations of an SRRP.

problems is the *topological fluidity* possessed by the SRRP (Topological fluidity is the ability to defer detailed routing until the overall routability is considered). The SRRP has also been studied under some additional constraints. Some of the restricted forms of the SRRP are :

- SRRP with prescribed street capacity: Here there is a limit (say two) on the number of allowed tracks in upper and lower streets. This SRRP does find applications in Printed Circuit Board routing, but it is not practical for VLSI routing problems.
- SRRP without crossovers: Here the aim is to produce a crossover free realisation for the SRR problem, even though it may not have optimal track congestion. A crossover free solution is not possible for all problems.
- SRRP with crossover bound  $K$ : This limits the maximum number of wires that can change over from one street to another in any of the inter-node spaces. The number of crossovers on each net is unrestricted as long as none of the inter-node spaces has more than  $K$  crossovers.

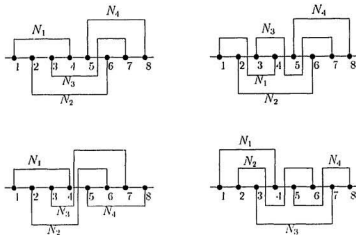


Figure 3.3: Examples of crossovers of different types.

### 3.1.1 Origins and developments of the SRRP

The SRRP was one of the earliest routing problems and it was introduced by So [So74], as a subproblem that arises in solving a general multilayer routing problem by a systematic decomposition into a number of independent single row single layer problems. This approach involves five steps, namely via assignment, linear placement of via columns, layering (also called via minimisation), single row routing and finally via elimination. Subsequently, algorithms with necessary and sufficient conditions to minimise the number of tracks were proposed by Ting *et al.* [Ting76]. Kuh *et al.* [Kuh79] proposed the *Interval Graph* (IG) representation of the SRRP. They also developed necessary and sufficient conditions for an optimum realisation, although they did not present an algorithm for SRRP based on these conditions. Raghavan *et al.* [Ragh82] proposed an algorithm based on *net order enumeration* technique which however, was not practical for large street capacities. Raghavan *et al.* [Ragh83] considered SRRP with backmoves, analysed SRRP with prescribed

street capacity and also discussed an algorithm to generate a crossover free solution, if one exists. Tarn *et al.* [Tarn84] utilised the conditions for optimum routing developed by [Kuh79] and proposed an efficient algorithm based on a simple heuristic, which produces an optimal solution for many problems. Raghavan *et al.* [Ragh84] reported detailed results on the computational complexity of the SRRP. Han & Sahni [Han84], modified the basic enumeration technique given in [Ragh82] for special cases of street capacity and later extended their work to deal with arbitrary street capacity [Han85]. None of the algorithms cited above paid any detailed attention to crossover minimisation. However, Du *et al.* [Du87] modified the net order enumeration technique of [Ragh82], taking into account the crossover bound as well. Du and Liu [DuLi87] extended the basic heuristic algorithm of Tarn *et al.* [Tarn84] by taking into account the *grouping effect*. Bhattacharya *et al.* [Bhat88] present a graph theoretic approach to solving the SRR Problem. Quite recently, Sherwani *et al.* [Sher89] have modified the algorithm in [DuLi87] taking into account the *clique intersection*.

Before describing the existing algorithms for solving the SRRP some more terminology is required. The interval graph representation [Kuh79] (also referred to as interval representation) of an SRRP is a set of  $m$  horizontal intervals representing the  $m$  nets together with an order, where  $m!$  such orders are possible. Figure 3.4.a illustrates one of the  $m!$  possible Interval representations for a given SRRP. The interval representation is a conceptual realisation for which there exists a corresponding unique physical realisation which can be obtained by a simple procedure. First a reference line is drawn which connects the nodes in succession from left to right. If the reference line is straightened out, then the  $m$  horizontal interval lines are mapped topologically into vertical and horizontal paths. Nets and portions of nets which lie above (below) the reference line are mapped into paths in the upper (lower) street. Figure 3.4.b depicts the physical realisation corresponding to Figure 3.4.a. Using the interval representation of an SRRP, it is easy to identify



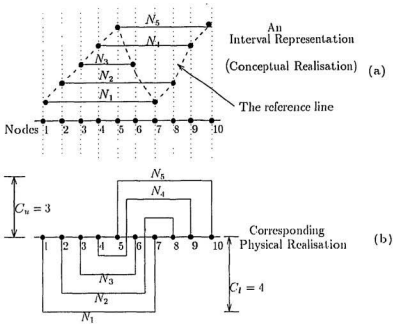


Figure 3.4: An interval representation and physical realisation of an SRRP.

the crossovers in the corresponding realisation. A crossover results whenever the reference line cuts a net in the interval representation. Also if there are a maximum of  $m$  nets, then the maximum crossover bound  $K_{max}$  is obviously  $(m - 2)$  [Du87].

Another important concept often used in an SRRP is the *cut number*. Each node and net has a cut number associated with it, which can be found from the interval representation. Consider any interval representation of an SRRP and draw an imaginary vertical line, perpendicular to the node axis, at every node. Then the number of nets, excluding the net to which the node belongs, which are cut by the vertical line at node  $i$ , is called the cut number  $C_i$ , for the node  $i$ .  $C_{iu}$  and  $C_{il}$  are the cut numbers for the node  $i$  in the upper and lower streets respectively. Clearly, the total cut number for the node  $i$  given by  $C_i = C_{iu} + C_{il}$

is an invariant property for a given SRR problem.  $C_{iu}$  and  $C_{il}$  obviously depend on the order of nets in the IG representation. With these definitions, the upper (lower) street congestion  $C_{us}$  ( $C_{ls}$ ) can be found as  $C_{us}(C_{ls}) = \max \{ C_{iu}(C_{il}) \}$  where  $1 \leq i \leq n$ . The overall congestion  $Q$  is  $\max \{ C_{us}, C_{ls} \}$ .

The cut number of a net  $N_i$ , denoted as  $q_i$ , is the maximum of the cut numbers of all the nodes which belong to the net  $N_i$ . Let the netlist  $L$  be partitioned into two sublists  $L_1$  and  $L_2$  such that  $L_1 \cap L_2 = \Phi$  and  $L_1 \cup L_2 = L$ . Then the internal cut number  $iq_j$  [Tarn84] of the net  $N_j$  in  $L$  with respect to  $L_1$  is defined as the cut number of  $N_j$  in  $L_1$ . Thus, the internal cut number of a net is similar to the total cut number of the net, except that only a subset of the entire netlist is considered. The residual cut number  $rq_j$  of the net  $N_j$  in  $L$  with respect to  $L_1$  is defined as the cut number of  $N_j$  in  $L_2 \cup \{N_j\}$ .

## 3.2 A taxonomy of algorithms for the SRRP

Because of the crucial role played by the SRR problem, attempts have been made to develop optimum<sup>1</sup> algorithms to solve the SRRP. Although the SRRP is a simple and well defined subproblem of the general routing problem, it is also NP-complete [Ragh84]. This has led to the search for efficient heuristic algorithms to solve the SRRP. What differentiates one interval representation from another is the ordering of nets. Thus, any algorithm to solve the SRRP should output an optimal ordering of nets which can then be translated into an optimally congested realisation.

The algorithms that have been reported so far for solving the SRRP can be grouped under two main families. The first family consists of the heuristic algorithm proposed by Tarn *et al.* [Tarn84], and its modified versions described

---

<sup>1</sup>Unless otherwise mentioned *optimality* means minimum track congestion in both streets.

by Du *et al.* [DuLi87] and Sherwani *et al.* [Sher89]. The second family is based on the net order enumeration algorithm proposed by Raghavan *et al.* [Ragh82] and its modified versions reported by Han and Sahni [Han84, Han85] and Du *et al.* [Du87].

### 3.2.1 Heuristic algorithms for the SRRP

#### Tarng's algorithm [Tarn84]

Since the SRRP is NP-complete [Ragh84], it is justifiable to look for efficient algorithms based on simple heuristics to solve the SRRP. Tarng *et al.* [Tarn84] proposed one such algorithm which is easy to code and often leads to optimal solutions. Basically, the algorithm assigns nets with large cut numbers close to the axis and the nets with small cut numbers further away from the axis. Thus, as one proceeds away from the node axis, in either direction, the nets will have non-increasing cut numbers. In order to achieve such an assignment, the nets are first grouped into *zones*. Nets within each zone are then ordered to get an overall ordering of nets which is then used to place the nets on tracks. Tarng's algorithm is now discussed in more detail.

Given an instance of an SRRP described by a netlist  $L$ , the node cut numbers and the net cut numbers are first determined; then the nets are partitioned into *zones* based on their internal cut numbers. Let  $q_{max}$  be the maximum net cut number. Then all nets with  $q = q_{max}$  are assigned to zone  $Z_0$ , all nets with  $q = q_{max} - 1$  are assigned to zone  $Z_1$  and so on until all the nets are exhausted. This step is referred to as *zone allotting*. Such zone allotment helps us to treat nets of the same cut number as a unit. For example, the SRRP illustrated in Figure 3.5 has  $q_{max} = 3$  and so there are 3 zones,  $Z_0 = \{1, 4, 5, 6, 9\}$ ,  $Z_1 = \{2, 3, 8\}$  and  $Z_2 = \{7\}$ , where the numbers within braces identify nets. In order to achieve the

goal of keeping the nets with large cut numbers close to the axis, it is clear that nets of  $Z_0$  should be closer to the node axis than nets of  $Z_1$  and so on.

The next step, called *net ordering*, is to find the ordering of nets within each zone. First the internal and residual cut number of each net in a zone  $Z_j$  with respect to  $L_1 = \{Z_0 \cup Z_1 \cdots \cup Z_j\}$  is found. Nets in a zone are then ordered in decreasing order of internal cut number. If two nets have the same internal cut number then they are ordered based on a decreasing residual cut number. It may happen that some nets have both the internal and residual cut number the same and in that case the relative ordering of such nets is arbitrary. For example, the order of nets shown in Figure 3.5, at this step would be  $Z_1 = \{5, (1, 4, 6, 9)\}$ ,  $Z_2 = \{(2, 3), 8\}$  and  $Z_3 = \{7\}$ , where the nets within ( ) can have an arbitrary order. Let the order be  $\{5, 1, 4, 6, 9\}$ ,  $\{3, 2, 8\}$  and  $\{7\}$ .

The last step is *track-assignment* in which the ordered nets from the zone  $Z_0, Z_1 \cdots$  are taken and assigned to tracks to get a conceptual realisation. First nets from  $Z_0$  are taken and put in fictitious track  $T_0$  or any other track  $T_j$  such that  $\|j\|$ , the absolute value of  $j$ , is minimised.  $T_0$  is tried first, then  $T_{+1}$  or  $T_{-1}$  is used and so on. Of course, only non-overlapping nets can be assigned to the same track. After finishing the assignment of all nets from  $Z_0$ , a similar procedure is followed for the nets of other zones but with the restriction that the nets of a new zone must occupy only those tracks which are not inner to the tracks used so far. This means that the nets of the new zone can share the outermost tracks already used, but may not use any interior tracks. For example, the  $Z_0$  nets in the example of Figure 3.5 are allotted to tracks  $T_0$ ,  $T_{-1}$  and  $T_{+1}$ . Net  $N_3$  can occupy the track  $T_{-1}$ . The above process results in a set of tracks with net assigned to it, and this conceptual realisation can be converted to a unique layout (realisation), by drawing the reference line and by topological mapping.

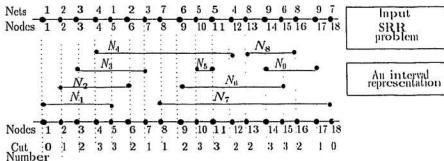
From the description of Tarng's algorithm some properties can be deduced; their implications will be discussed in Chapter 5.

- *Property 1* : Whenever there is a choice between using track  $T_{+j}$  or  $T_{-j}$  for a net, the choice is irrelevant. The goal of Tarng's algorithm is to minimise the distance of the new fictitious track from track  $T_0$  and it is immaterial whether the new track is in the upwards or downwards direction.
- *Property 2* : Within each zone, if some nets have the same internal as well as residual cut numbers, then the order of assigning them to tracks is arbitrary.

### Du's algorithm [DuLi87]

Du's algorithm is basically an improved version of Tarng's algorithm. Tarng's algorithm being a heuristic algorithm it is not expected to produce an optimal solution for all cases. Du *et al.* analysed the cases for which Tarng's algorithm fails to produce an optimal solution and introduced the concept of *groups*. The netlist is divided into subsets, such that nets in each subset satisfy the *grouping* property. Maximal set of nets which cut across one or more common node form a group, i.e., there is at least one vertical line which cuts all the nets of a group. The concept of grouping is illustrated in Figure 3.6. Here the total netlist  $L = \{N_1 \cdots N_7\}$  can be partitioned into two groups,  $g_1 = \{N_1, N_2, N_3, N_4\}$  and  $g_2 = \{N_3, N_5, N_6, N_7\}$ , with net  $N_3$  being common to both groups.

Du *et al.* have proved that Tarng's heuristic principle always produces an optimally congested solution as long as the input nets form a *single group*. In the case of *multi-group* nets an optimal solution cannot be guaranteed. Du *et al.* proposed that a multi-group SRR problem be broken in to a number of single group SRR problems. That is, groups should be identified and then each group routed independently using Tarng's approach, resulting in an ordering of the nets within



Nets	Cut-Number	Zone
$N_1, N_4, N_5, N_6, N_9$	3	$Z_0$
$N_2, N_3, N_8$	2	$Z_1$
$N_7$	1	$Z_2$

$$Z_0 = \{5, (1,4,6,9)\}$$

$$Z_1 = \{(2,3), 8\}$$

$$Z_2 = \{7\}$$

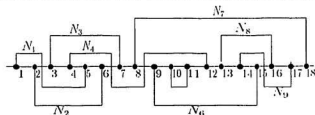
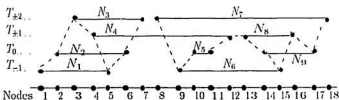


Figure 3.5: Routing a SRR problem with Targ's algorithm.

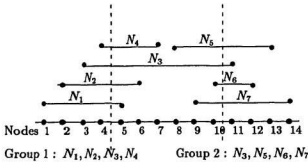


Figure 3.6: Concept of groups

each group. Then the groups should be merged so as to arrive at an overall net order while preserving the relative order of nets within each group. Such an overall ordering is not always possible, especially when there are many nets which belong to more than one group; i.e., when there is a lot of interaction between the groups. Du *et al.* have shown that if such an ordering were found, then it would be optimal. Clearly, an optimal solution is produced for those cases in which no more than two nets are common for adjacent groups. Consider the two group SRR problem shown in Figure 3.6. A possible realisation using Tarng's algorithm is shown in Figure 3.7 with a congestion of 3. On the other hand, if Du's concept of grouping is applied, and the solution of the individual groups are merged then the final SRR realisation has an overall optimal congestion of two as shown in Figure 3.8.

A few remarks on property 1 and property 2 introduced above are in order. Property 1 and property 2 of Tarng's algorithm give some choice in the *track assign* phase and result in different ordering of the nets. When the nets form a single group such orderings translate into optimally congested realisations. However, in the case of multi-group nets, not all the orders which satisfy property 1 and property 2 lead to street optimality. For both single and multi-group SRR problems, different orderings which satisfy property 1 and property 2 lead to different crossover counts.

Thus, to get a solution with fewer crossovers, the choices made possible by property 1 and property 2 have to be restricted by additional constraints. This topic will be addressed again in Chapter 5.

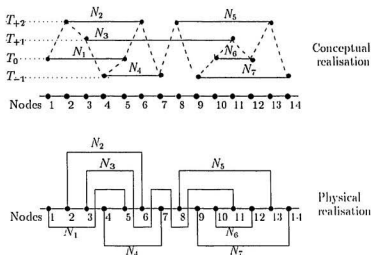


Figure 3.7: A 2-group SRR problem routed with Tarng's algorithm

### Sherwani's algorithm [Sher89]

Sherwani *et al.* [Sher89] have reported yet another algorithm which is basically a hybrid of Tarng's algorithm and Du's algorithm. Since Tarng's algorithm produces an optimal solution for single group nets and for multi-group nets, if the interaction between adjacent groups is quite high, then the problem is similar to a single group case and Tarng's algorithm is still likely to succeed. Du's algorithm considers the net orders in each group based on the local cut numbers and it is increasingly difficult to find an overall order without affecting the relative order within each group, when the interaction between groups is quite high. Thus, Tarng's algorithm works well with single group nets and highly interacting multi-group nets while



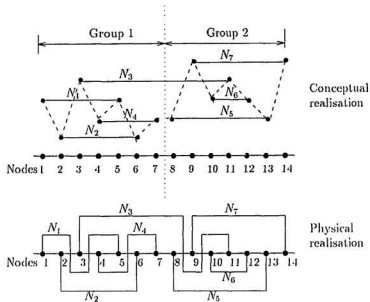


Figure 1.8: A 2-group SRR problem routed with Du's algorithm

Du's algorithm works well with loosely coupled multi-group nets.

Sherwani *et al.* have exploited this observation and have described a hybrid scheme. First the groups are identified and if there is high interaction between two adjacent groups, then it is neither useful nor necessary to keep them as two groups and hence they are merged into a *pseudo group*. For each such pseudo group, Tarnag's algorithm is applied to find a local order within the pseudo groups and then an overall order is attempted. Thus, Sherwani's approach performs like Tarnag's algorithm when the interaction is high between all adjacent groups and performs like Du's algorithm when all the adjacent groups are loosely coupled. However, in realistic routing problems Sherwani's algorithm is likely to succeed better than either Tarnag's or Du's algorithm. What is high or low interaction and when the groups should be merged into pseudo groups is left to the decision of the

user to be determined empirically.

Sherwani *et al.* have also brought out some interesting graph theoretical interpretations of groups and their interactions. Given an SRRP, construct an *Interval Graph* which is an intersection graph of the intervals representing the nets. The nodes of the interval graph represent nets of the SRRP and there is an edge from node  $i$  to node  $j$  of the graph if net  $N_i$  and net  $N_j$  intersect in the interval representation. There are two possible types of intersections, namely, *overlap* and *containment*. Two nets are said to have a containment type intersection if one net is contained totally within the other, while on the other hand, if one net is not contained in the other, but if they have a common horizontal stretch, then the two nets are said to overlap, as illustrated in Figure 3.9.

Since any two nets of an SRRP must be non-intersecting or overlapping or containing, three types of intersection graphs are possible for an SRR Problem. The intersection graph which captures only the containment relations between nets is called *containment graph*  $G_c$ . Similarly, the intersection graph which captures only the overlap relations between nets is called *overlap graph*  $G_o$ . The intersection graph which captures both overlap and containment relations is called *overall interval graph*  $G_I$ . The maximal cliques of the overall interval graph  $G_I$ , can be identified as the groups of Du and Liu [Dul87]. The number of common nets between adjacent groups will correspond to the number of common nets between the cliques and this is called clique intersection.

### 3.2.2 Enumeration algorithms for the SRRP

The first member of this family of algorithms was proposed by Raghavan *et al.* [Ragh82]. This approach is basically different from the Tarng's family of algorithms in that Raghavan *et al.* use an exhaustive or semi-exhaustive enumeration of all

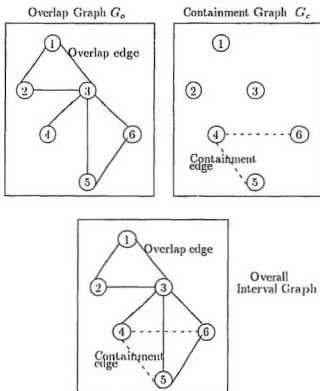
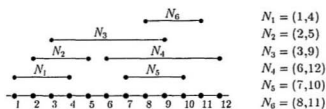


Figure 3.9: Interval graph representation and groups of an SRR problem

the possible orders in which nets can arrive at a node. This family of algorithms finds such feasible orders by imposing restrictions on the street capacity, crossover bound, *etc.*

Before proceeding further more terminology must be introduced. Any node can be classified as a *Begin (B)* node or a *Middle (M)* node or an *End (E)* node. A node is of type *B* if a net starts from that node. That is, the node is the leftmost node of a net. Similarly, a node is of type *E* if a net ends there, *i.e.*, that node is the rightmost node of that net. Nodes of a net which are neither *B* nor *E* are called *M* nodes, representing those nodes which lie between *B* and *E* and are touch points for that net.

#### **Raghavan's net order enumeration algorithm [Ragh82]**

Raghavan's [Ragh82] algorithm progresses from left to right and considers, in a systematic way, all the possible orders by which the nets can arrive at the nodes. Basically, given the orders of nets arriving at node  $i$  and the restriction on the street capacity, the feasible orderings that can arrive at node  $(i + 1)$  are generated, the criterion for feasibility being non-overflow of the prescribed street capacity. Each of the feasible incoming orders at node  $(i + 1)$  can then be used to generate feasible orders at node  $(i + 2)$  and so on. If there are no feasible incoming orders at any of the nodes, the method fails and no realisation of that netlist within the prescribed street capacity is possible. On the other hand, if the last node is reached, then the algorithm succeeds and there are one or more possible realisations. Each of the back track paths from the last node to the first node gives a net order for a feasible realisation. Each such realisation will satisfy the prescribed street capacity.

Let  $N_1, N_2, N_3 \dots N_k$  be one of the many feasible orders of nets arriving at node  $i$  as shown in Figure 3.10.a. Assume, for simplicity, that the prescribed

upper and lower street capacities are the same, say  $k$ . The goal is to find all the feasible orders that can arrive at node  $(i + 1)$ . Each such feasible order should not make either the upper or lower street overflow in the interval  $\langle i, i + 1 \rangle$ . Further, the node  $(i + 1)$  could be a  $B$  or  $M$  or  $E$  type. Assuming a  $B$  type node, the valid and forbidden levels where the net whose  $B$  node is  $i + 1$  can be placed are also shown in Figure 3.10.b.

The main drawback of the exhaustive enumeration technique described above is that its time complexity is  $\{O(k! \cdot n \cdot k \cdot \log k)\}$  and as  $k$  becomes larger the method becomes impractical. Even for small values of  $k$ , orders which would eventually lead to non-feasible orders are generated and propagated up to the failure node. This led to attempts to develop better techniques to reduce the number of feasible orders by imposing stronger or additional constraints. Han *et al.* [Han84, Han85] imposed stringent restrictions on street capacity and prescribed them to be low values (say 2 or 3), to eliminate a lot of orders as not feasible while Du *et al.* [Du87] use the *crossover bound* ( $K$ ) as an additional constraint. These approaches make a semi-exhaustive search for the feasible orders and are thus better than the exhaustive approach of Raghavan *et al.* [Ragh82].

### Han & Sahni's improved enumeration algorithms

Han *et al.* [Han84] developed  $O(n)$  algorithms for the cases when  $k_u, k_l \leq 3$  as well as for the case when  $k_l = 1$  and  $k_u$  is arbitrary. The case for  $k_u, k_l \leq 3$  was actually solved by developing algorithms for the subcases  $k_u, k_l = 2$ ,  $k_u, k_l = 3$  and  $k_u = 3, k_l = 2$ . These algorithms together with that for  $k_l = 1$  and  $k_u$  arbitrary, cover all the possibilities for  $k_u, k_l \leq 3$ . The case of  $k_l = 0$  and  $k_u$  arbitrary is trivial and can be solved in linear time. The stringent limits on the street capacities drastically cut down the explosion of feasible orders resulting in a faster algorithm. Further the algorithm considers only symmetric permutations {e.g. order  $abcd$  is

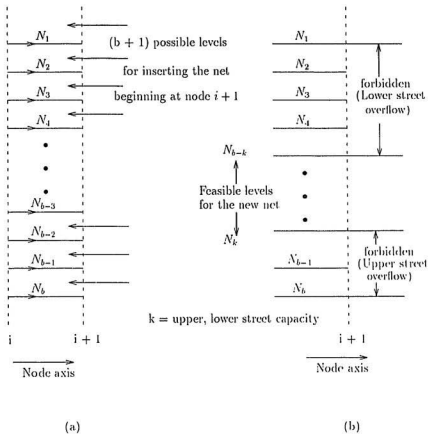


Figure 3.10: Feasible orders generation for a B node.

symmetric to order *dcb*a, because the corresponding realisations are simply mirror images of each other about the node axis). Han *et al.* [Han85] developed a fast algorithm for the case of arbitrary  $k_u$  and  $k_l$ . Here, the explosion of feasible orders is controlled by some more efficient permutation reduction techniques which eliminate orders (which would eventually become not feasible) at the early stages of the execution of the algorithm.

### Du's algorithm with Crossover bound

Du *et al.* [Du87] considered the SRR Problem with prescribed street capacities and crossover bound, where the crossover bound is exploited as an additional constraint to control the explosion of feasible orders. With a small crossover bound (say 2 or 3), many permutations are ruled at the early stages of the algorithm execution and need not be generated at all. Du *et al.* have also described a systematic procedure to construct a netlist such that any of the realisations will need at least as much as a prescribed crossover bound. In the absence of the crossover bound, the maximum number of nets that can crossover is limited only by the street capacity. However, stipulating a much lower value for the crossover bound  $K$  drastically reduces the feasible orders.

### 3.2.3 Concluding remarks

It is worthwhile to make a comparison of the two families of algorithms that were discussed, namely, enumeration algorithms and heuristic algorithms. The *enumeration algorithms*, whether exhaustive or semi-exhaustive, suffer from time complexity and are not practical for large problems. In general, enumeration algorithms do not outperform heuristic algorithms and do not provide a better or faster solution for some classes of problems. They have limited applicability, but are quite useful

in cases of low street capacities. It is quite evident that the *heuristic family* of algorithms is a better choice in terms of their computational complexity. They produce reasonable solutions, if not optimal, for many cases, and there is always scope to find a simpler or better heuristic for the same problem as well as to evolve a new heuristic for a related problem. For example, none of the reported heuristic algorithms have considered crossover minimisation as the optimisation criterion. The problem of minimum crossover routing is important in the development of a channel router based on the SRT approach, so developing a heuristic algorithm for minimum crossover routing is an interesting option, which will be addressed in Chapter 6.



## Chapter 4

# The Single Row Transformation (SRT) Approach

In this chapter the Single Row Transformation (SRT) approach to VLSI routing proposed by Wong *et al.* [Wong88] is introduced. The SRT approach requires three steps: the Forward Transformation (FT), SRR solving, and the Backward Transformation (BT). An analysis of the work reported in [Wong88] reveals that although SRT is an apparently simple approach, it has hidden problems which are completely unaddressed in the superficial treatment presented in [Wong88]. In particular, the crossovers in the SRR solution pose problems in the BT step, making an algorithmic procedure for BT difficult. These problems are illustrated using suitable examples and a two step approach to crossover management, involving crossover reduction and crossover handling is proposed. The difficulty in managing crossovers leads to the conclusion that crossover minimisation is essential for the SRT based channel router. The concept of *pseudo stretches* in SRR is proposed and its applications are indicated. Since crossovers play a crucial role in SRT based routing, a new taxonomy of SRR problems which is suitable to their application in SRT environment is proposed and discussed.

The SRT approach to VLSI routing attempts to solve a given routing problem by solving an SRR problem. The *Forward Transformation* step converts the given routing problem (say, a channel or switchbox) into an equivalent SRR problem, by rearranging the pins on the boundary of the input problem onto a single row. The *SRR solver* then produces a realisation of this SRR problem. Finally the *Backward Transformation* folds the SRR solution back into a channel solution. The steps of FT and BT are collectively referred to as *single row transformations* (SRT). Figure 4.1 illustrates SRT based routing of a simple channel. It is to be noted that the SRT approach is applicable not only to channel or switchbox routing but also to any closed polygonal routing domain. The success of the SRT approach is determined by the efficiency of its three steps. As discussed in [Wong88], the SRT approach to channel routing usually leads to solutions whose qualities are comparable to those produced by other channel routers.

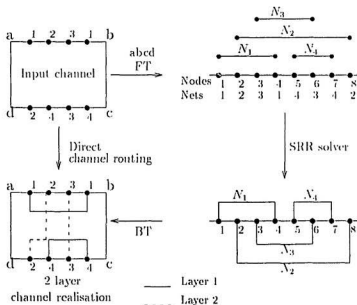


Figure 4.1: Channel routing by the SRT approach

## 4.1 Different FT-BT pairs

The FT and the corresponding BT are, in a sense, the inverse of each other. Whatever way the terminal order of the channel is altered in the FT step, it must be restored in the BT step. Thus, the *folding back* operation in BT, which maps the SRR layout and ensures terminal order restoration, becomes complex for those FTs which do not preserve the adjacency of channel terminals. The folding back operation is also complicated by the presence of crossovers in the SRR layout, as explained in a subsequent section. Thus, although different FT-BT pairs are possible, the practical choices are restricted by the complexity of the BT step. In fact, the BT is the crux of the whole SRT approach and the FT and the SRR solver should facilitate the BT. Given a channel with  $n$  pins on either side, a total of  $2n!$  FT-BT pairs are possible. In the following section, different FT-BT pairs are classified and presented.

### Edge preserving transformations

These transformations preserve the order of terminals in both the top and the bottom sides of the channel during the FT operation. That is, the top and bottom edges of the channel are treated as a single unit and placed in different ways on a single row. Consider a channel  $abcd$ , whose top edge is  $ab$  and bottom edge is  $cd$ , as shown in Figure 4.2. Then there are eight possible transformations which preserve the order in each edge of the channel while forming the SRR problem as listed in Figure 4.2. Out of these eight transformations, only four are basic and the other four are mirror versions. Let the basic set be  $abcd$ ,  $abdc$ ,  $dcab$  and  $cdab$ . Out of these four, only  $abcd$  and  $cdab$  are obtained by a straightforward opening out of the channel by swinging one edge in line with the other. This leads to a trivial terminal order restoration during the fold back process of the BT. Further, if the

SRR layout is crossover free then the wires can be deformed easily into a channel without vias, by assigning all the nets in the upper street to one layer and all the nets in the lower street to the opposite layer. The channel solution is guaranteed and can be obtained within a maximum of twice the SRR congestion. Such a bound cannot be stated when there are crossovers. The crossovers in the SRR solution would translate into vias in the channel solution. Apart from its simplicity and practicality the *abcd* FT results in a number of comparable properties between the channel and the SRR problem. For example, the *circle graph*  $G_c$  of the channel is identical to the *overlap graph*  $G_o$  of the SRR problem and the via minimisation of channels is quite similar to the crossover minimisation of SRR problems. The detailed operation involved in the *abcd* type FT-BT pair is illustrated in Figure 4.3.

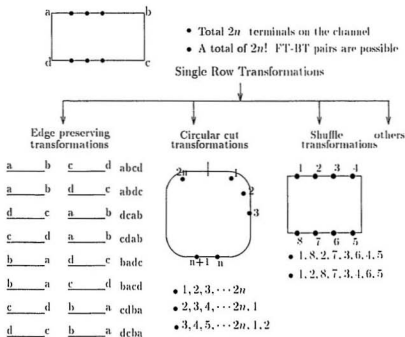


Figure 4.2: Different FT-BT pairs for the SRT based routing

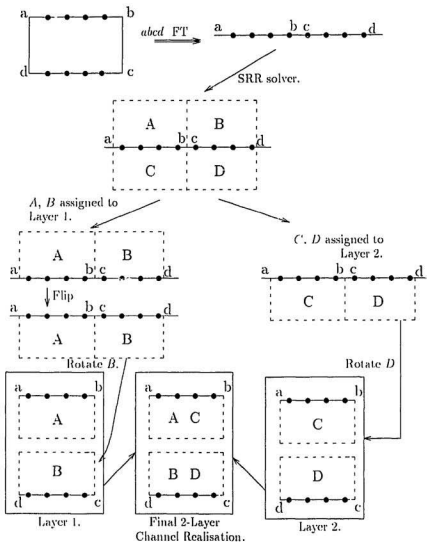


Figure 4.3: Steps of the *abcd* type SRT

## Circular cut transformations

Another approach is to imagine the  $2n$  terminals of the channel to be located on the circumference of a circle. Then the circle can be cut open at any one of the  $2n$  points to form a single row of nodes, as illustrated in Figure 4.2. Again at each cut the terminals can be visited in either a clockwise or a counterclockwise fashion, leading to mirror versions of the SRR problem. Some of the edge preserving transformations (e.g.,  $abcd$ ) can be visualised as a circular cut transformation as well. All the  $2n$  circular cut transformations have the same *overlap graph*, identical to the circle graph of the channel, because two nets which cross in the channel would overlap, no matter at what point the circle is cut open to form the SRR problem. Two nets which do not cross in the channel (*i.e.*, containing or non-intersecting nets) could translate into containing nets or non-intersecting nets depending upon the cut point, but a containment relation in channel cannot become an overlapping relation in the SRR problem. Thus, each cross in the channel is reflected as a overlap in the SRR problem.

## Shuffle transformations

Here the terminals in the top and bottom edges are mixed when the SRR problem is formed. A number of shuffle transformations (permutations) are possible, each destroying the channel terminal adjacency to a different extent. While folding back SRR layouts obtained by shuffle transformations two wires that do not cross in the SRR layout may cross, leading to difficulties which must be resolved by the BT. Thus, in general, such transformations call for additional work during the BT step.

Based on these considerations it is clear that the  $abcd$  is a good choice in terms of its simplicity and practicality. Of course, different FT-BT pairs work differently for a given channel problem, leading to different SRR problems. The

SRR solver could then produce solutions which vary in the number of crossovers resulting in simpler or more difficult SRR layouts. Again the apparent simplicity of the SRR solution (*i.e.*, fewer crossovers) is unrealistic because the task of terminal order restoration would lead to conflicts when the SRR layout is folded back, making the BT's success highly input problem dependent. It is not known yet whether a single FT-BT pair is optimal for a family of input problems. Thus, the question of finding an optimal FT-BT pair which is likely to succeed on many cases is an open problem.

## 4.2 Some remarks on Wong's paper

The paper due to Wong *et al.* [Wong88] was the first paper<sup>1</sup> that discussed the SRT based approach to VLSI channel routing. Although Wong's paper proposed this novel idea, the treatment presented therein was very superficial. The discussions were based on a few examples. Although some problems with BT were indicated, no algorithmic procedure was presented. Further, their observation that the SRT approach leads to fewer vias compared to the other approaches is not correct. In the examples shown in their paper, the fact that fewer vias resulted in the case of SRT based routing has nothing to do with the SRT routing itself, but is due to their implicit assumption that all terminals are accessible in both layers. This is not assumed in the compared routers, namely, Weaver and Yoshimura & Kuh's algorithms. If the terminals are accessible in only one layer then the SRT approach would have also resulted in nearly the same number of vias as the other two methods. Another major drawback in their presentation is that the problems posed by crossovers (in the SRR solution) with respect to the Backward Transformation are completely unaddressed. In fact, the presence of crossovers in the SRR solution

---

<sup>1</sup>In fact, the only paper so far, to the best of the author's knowledge.

is the main reason for the complexity of the Backward Transformation. In the absence of crossovers, the BT is a straightforward operation and a final channel solution is guaranteed, which is not the case in the presence of crossovers. This calls for work on the minimum crossover routing of SRR problems which has been discussed very little in previous literature. None of the existing algorithms is directly suitable as SRR solvers in the SRT based environment, because they were designed for optimising the track count and not the number of crossovers. Consequently, the suggestion by Wong *et al.* that Tarnig's algorithm [Tarn84] can be used as the SRR solver is not the best choice as it would not lead to minimum crossovers needed for the SRT environment<sup>2</sup>. Thus, it is necessary to design new algorithms or modify existing algorithms for crossover minimisation. Even with optimal algorithms still there could be crossovers, because not all SRR problems have a crossover free solution. This calls for ways to handle these unavoidable crossovers to make a practical BT. Such handling procedures are complicated by the number as well as the nature of the crossovers, as will be explained in the next section. Thus, although SRT based routing has potential applications, its practicality remained to be proven and more work was needed to explore the unsolved issues. This is the main motivation behind choosing SRT based VLSI routing as the topic for this thesis.

### 4.3 Problems posed by crossovers

Under the *abcd* FT-BT, the cases which have no crossovers in the SRR solution are easy to fold back and the terminal order restoration is trivial. This is because the SRR layout can be converted back to the channel in a way exactly opposite to the way in which the channel was folded open to form the SRR problem. All

---

<sup>2</sup>As explained in chapter 3, Tarnig's algorithm does not necessarily produce a crossover free solution even if one exists.



the upper street nets of SRR layout can be translated into nets of layer 1 and all the lower street nets can be translated into nets of layer 2 of the channel as shown in Figure 4.1. Such a simple folding back is not possible, however, in the presence of crossovers, because of a basic difference between the channel model and the SRR model. While crossovers are allowed in the internode space in the SRR model, vias are to be located only at grid points in grid based channel routing. However, in a direct mapping of SRR layout onto the channel, the crossovers of SRR would translate into vias located at mid grid points and this calls for grid alignment of vias. Therefore, the presence of crossovers results in an additional and potentially complex process. Here the BT becomes complex because the grid alignment of vias may need re-routing of nets without vias. Instead of aligning the vias, an alternative is to break the crossover in the SRR solution such that the split segments of the crossing net would contact through a grid aligned via after the fold back. However, such an alignment may not be guaranteed for all problems and types of crossovers. Some crossover alignment methods will be illustrated now through some examples.

Figure 4.4 illustrates a channel problem along with a realisation with one crossover for the SRR problem obtained by the *abcd* FT. To grid align the crossover the logical choice is to use the nearest grids, that is the columns at node 2 or 3. However, this is not possible because other nets have already used this grid. An alternative is to translate the crossover on  $N_1$  as a split of net  $N_2$  as shown. This layout after fold back results in a layout with a grid aligned via as shown. However, a careful observation reveals that the via is not needed at all. In this case the crossover alignment process could have been avoided altogether if one had started with a crossover free realisation. It is to be noted that the alignment process also results in an additional track  $T_{+2}$  which is not there in the original SRR layout. Thus, it is clear that whenever a crossover free solution is possible, it is desirable that the SRR solver produce it. This is the focus of Chapters 5 and 6.

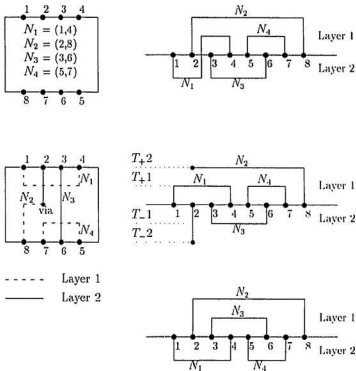
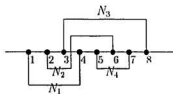


Figure 4.4: An example of an easily alignable crossover

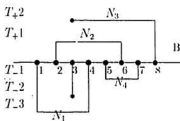
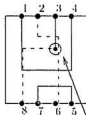
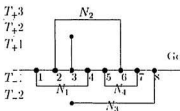
A more complicated example is shown in Figure 4.5. The SRR problem obtained by *abcd* transformation of the channel has no crossover free solution because its overlap graph is not bipartite and needs at least one crossover for any of its realisations. Consider the SRR solution with one crossover realisation as shown Figure 4.5. Two methods of aligning the crossover of net  $N_2$  is presented, both of which break the net  $N_3$ , but in different ways. However, upon folding back, one of them translates into a legal channel layout whereas the other has the problem of net overlap leading to an illegal channel. Of course, such a net overlap could have been avoided by having a detour on net  $N_2$ , as shown. Thus, legal SRR layouts can translate into illegal channel layouts if care is not taken. The crossover management process is highly case dependent and could turn out to be a difficult and sometimes an impossible task, without adding an extra column. A naive Backward Transformation usually results in redundant vias as well extra tracks (for example, if the redundant detours in channel are not removed). Further, the BT should be non-geometric. That is, the BT should not attempt to transform the SRR layout to the corresponding channel layout on a segment by segment basis. When crossovers are present either due to the poor performance of the SRR algorithm or to the inherent nature of the input channel problem, general crossover handling techniques to manage any kind of crossover are needed and these issues will be addressed in Chapter 7 on Backward Transformation.

## 4.4 Crossover management techniques

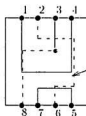
This section describes techniques to manage the crossovers in the SRR solution. A two step approach to crossover management is proposed, the steps being *crossover reduction* and *crossover handling*. The crossover reduction step is the process which



At least one crossover necessary  
for any SRR realisation



Net overlap problem



Detour of net  $N_2$   
to avoid the net overlap.

Figure 4.5: A crossover which needs a careful alignment

tries to minimise<sup>3</sup> the number of crossovers in the SRR solution, and to obtain a crossover free solution, if any. The task of minimum crossover routing is clearly analogous to the Unconstrained (Topological) via minimisation of channels. The crossover handling step which manages whatever crossovers that remain after the reduction process would be the front end of the BT and will be discussed later in Chapter 7.

Since the existing SRR algorithms do not produce a minimum crossover routing, algorithms with integrated crossover reduction are needed for the SRT based router. Such algorithms can be obtained by suitable modifications to existing SRR algorithms or by devising totally new algorithms. Both approaches have been pursued in this thesis and the results are presented separately for crossover free routing (Chapter 5 on bipartite SRR) and for minimum crossover routing (Chapter 6 on non-bipartite SRR). An SRR algorithm with integrated reduction increases the quality of the SRR solution and results in fewer crossovers that need be handled by the BT.

## 4.5 Pseudo stretches in SRR

Consider a channel with 2-pin nets. Each such net is a *permutation* net or a *local* net. A permutation net has one pin on the top side of the channel and the other on the bottom side of the channel whereas, the local net has both pins on the same side of the channel. A permutation net can be a *left* (*L*), a *right* (*R*), or a *straight* (*S*) net where the classification depends on the direction of flow of the net from top side to bottom side. The *span* of the net corresponds to the horizontal stretch of the net in the channel. The span of straight nets is zero while those of others are

---

<sup>3</sup>Note that crossover minimisation in SRT framework does not necessarily mean absolute minimisation of crossovers, as will be discussed in chapter 6.

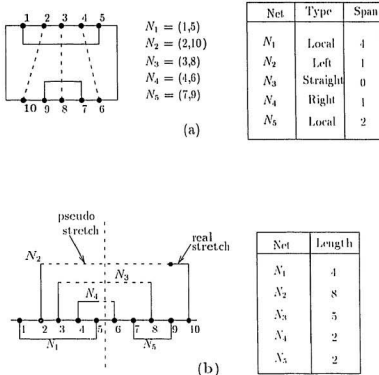


Figure 4.6: Translation of channel nets to SRR nets

non-zero. Examples of channel nets and their spans is illustrated in Figure 4.6.a.

When a net in the channel is transformed into a net in the SRR model using the *abcd* FT, the length of the net in the SRR domain is the same as its span only for the local nets and the SRR length of left, right and straight nets is greater than their spans in the channel, as illustrated in Figure 4.6.b. Thus, any permutation net (*L/R/S*) has a *pseudo stretch* in the SRR whose length is equal to the difference between the SRR length of the net and its channel span. Pseudo stretches in the SRR will translate into detours in the channel when the SRR solution is folded back. Detouring a net in the channel increases its width

and complicates the BT as well and hence is not desirable. Thus, assuming no detours are allowed in the channel, the pseudo stretches in the SRR model should not show up in the final channel. The stretch which is not pseudo will be referred to as real. The pseudo stretch of the nets in Figure 4.6.b is shown in dotted lines while the real stretch is shown in solid lines. Since a pseudo stretch of an SRR net is symmetrical about the central axis  $YY$ , the length of the pseudo stretch is always an odd number. The entire length of a straight net in the SRR is pseudo whereas left, right nets have a pseudo as well as a real stretch.

Apart from reducing the channel width, pseudo stretch removal has another important application, namely, in *crossover* management. Any crossover that occurs within the pseudo stretch of an SRR net will be called a *pseudo crossover* whereas crossovers which are not pseudo are real. Since pseudo stretches of SRR eventually do not show up in the channel the course of the pseudo stretch of an SRR net does not have any effect on its shape in the channel. For example, all the crossovers on a straight net can be collapsed into a single via and there is not a one-to-one translation of crossovers into vias. Thus, it turns out that any number of pseudo crossovers can be translated into a single via and this phenomenon is called *crossover collapsing*. This greatly aids the crossover handling process and will be discussed in Chapter 6 on non-bipartite SRR.

## 4.6 A Taxonomy of SRR problems

Historically the SRRP was formulated as a special subproblem by H.P. So [So74] in the systematic decomposition of Multilayer PCB routing problem into a number of independent single row single layer routing problems and in this application the conventional *street congestion* was the criterion for optimisation. However, for the application in SRT based routers crossover minimisation proves to be a better and

hence primary criterion for optimisation. Thus, in the light of this new application, a taxonomy of SRR problems is now proposed.

An SRR problem has a crossover free solution if its overlap graph  $G_o$  is bipartite, and needs at least one crossover in any of its solutions, if  $G_o$  is not bipartite. Thus, SRR problems can be classified into *Bipartite SRR* (BSRR) problems (which have a crossover free solution) and *Non-bipartite SRR* (NSRR) problems (which do not have any crossover free solution). The BSRR and the NSRR problems can be further subdivided based on the type of the original channel problem, namely, Permutation Channel or Mixed channel. In a Permutation Channel routing problem (PCRP) all the nets are 2-pin nets and each net has one pin on the top side and another on the bottom side of the channel. That is, only straight, left and right nets are allowed. Whereas, a Mixed Channel routing problem (MCRP), can have multipin nets as well as local nets in addition to the permutation (left, right and straight) nets. The SRR problem obtained from a PCRP by *abcd* FT is a PSRR problem. Similarly, the SRR problem obtained from an MCRP by *abcd* FT is an MSRR problem. The BSRR problem could belong to the Permutation or Mixed type, resulting in Permutation Bipartite SRR (PBSRR) or Mixed Bipartite SRR (MBSRR) problems respectively. In a similar way, an NSRR problem could be divided into Permutation Non-bipartite SRR (PNSRR) and Mixed Non-bipartite SRR (MNSRR) problems. This taxonomy is shown in Figure 4.7 and some examples have been also illustrated. This taxonomy is the basis of the subsequent sections of this thesis.



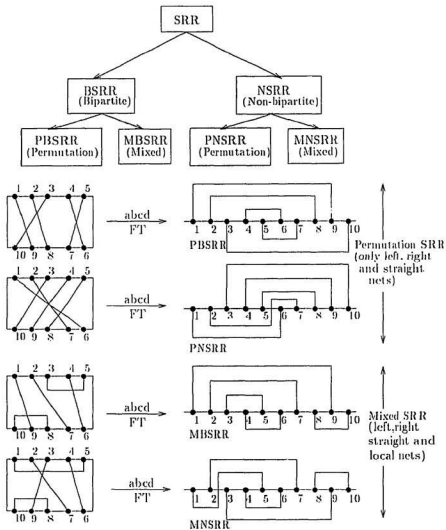


Figure 4.7: A taxonomy of SRR problems

## Chapter 5

# Bipartite Single Row Routing

In this Chapter the PBSRR and the MBSRR problems, which are the crossover free, or *bipartite*, cases of the PSRR (Permutation SRR) and the MSRR (Mixed SRR) problems respectively, are discussed. The goal is to find a crossover free solution, if the input SRR problem belongs to the PBSRR/MBSRR category. This Chapter begins with a brief overview of previous work in obtaining a crossover free solution of an SRR problem highlighting the different formulations of the *crossover free routability* problem. Then a new  $O(n)$  algorithm is proposed which finds a via free solution, if any, of a Permutation Channel Routing problem in terms of a crossover free solution of the corresponding PBSRR problem using the *box procedure* [Veng90]. Possible extensions of this algorithm are also discussed. Next, modifications to Tarug's algorithm [Tarn84] are proposed so that the modified algorithm *TARNG-MOD* would produce a crossover free solution, if any, for an SRRP. This approach is general and applicable to both PBSRR and MBSRR problems. This algorithm has been coded in the C language and details of the implementation will be presented in Chapter 8.

Crossover free routing is a restricted form of single row routing. Given a netlist, the objective is to determine a wire layout that minimises the congestion  $Q = \text{Max}\{C_u, C_l\}$ , subject to the additional constraint that no wire may switch from one street to another. It should be noted that the minimum street congestion achievable under the crossover free model, in general, is higher than the optimal congestion given by  $Q_o = \text{Max}\{q_{\min}, \lceil q_{\max}/2 \rceil\}$ . Due to the problems posed by crossovers, as explained in Chapter 4, priority should be given to crossover minimisation rather than track minimisation while routing the SRRP for the application in SRT based routers. Thus, a more congested SRR solution with no crossovers would normally be preferred to a less congested SRR solution with crossovers. Again, for a given SRRP there may be more than one crossover free solution, and the least congested of these is preferred. Note that, based on the definition of congestion  $Q$ , the solutions  $\{C_u = 2, C_l = 3\}$ ,  $\{C_u = 3, C_l = 2\}$  and  $\{C_u = 3, C_l = 3\}$  would all have the same congestion  $Q = 3$ , and all are equally acceptable.

Ting *et al.* [Ting76] gave algorithms and necessary and sufficient conditions for obtaining a legal realisation of an instance of an SRRP. In their paper, they showed that if the number of tracks available in each street is unlimited, then any set of nets is realisable, with the implicit assumption that there are no restrictions about crossovers. Kuh *et al.* [Kuh79] went one step further and have shown that in the absence of any restriction (on street capacity or crossovers) an optimally congested (*i.e.*,  $Q = Q_o$ ) solution is possible for any SRRP. However, with the restriction that crossovers between nodes are not allowed, these results no longer hold true; that is, not all net lists have a legal realisation. An example of an SRRP impossible to wire without a crossover is illustrated in Figure 5.1.a. Further, even if a netlist has one or more crossover free solutions, none of these may have the optimal congestion  $Q_o$ .

Thus, in connection with the crossover free routing of an SRRP, one is interested in answering two questions, that of *feasibility* (which has a *yes* or a *no*

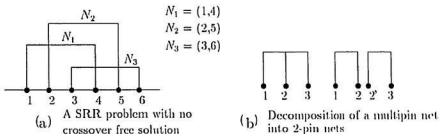


Figure 5.1: Some examples of SRRP nets.

answer) and of *realisation*, which describes which nets occupy which tracks in the layout. It is sufficient to know just the track number of each net in order to fully characterise an SRR layout without crossovers. Further, the restriction that wires cannot crossover allows one to decompose a multipin net as a sequence of 2 pin nets with the intermediate pins temporarily duplicated and later collapsed, the collapsing being possible because no crossover lies in between the pins to conflict with the collapsing. This is illustrated in Figure 5.1.b. Thus, in the following discussion all the nets are assumed to be 2-pin nets.

## 5.1 An overview of crossover free routing

The crossover free routing problem was of much interest in the early PCB technologies when the design rules did not permit any etch path between the adjacent pins of the dual-in-line IC packages. The earliest work which considered the problem was [Fost79], but no satisfactory solution for crossover free routing was proposed therein. Tsukiyama *et al.* [Tsuk80] have shown that finding a realisation that minimises the total number of crossovers is NP-complete. Tsui *et al.* [Tsui81] present necessary and sufficient conditions for routability of a set of two-pin nets to be wired in a single row without using vias. They also describe an algorithm to achieve a crossover free solution which has a time complexity of  $O(n \cdot 2^Q)$ , where  $n$  is the

total number of nodes and  $Q$  is the overall congestion ( $\max\{C_u, C_l\}$ ) of the SRR realisation. This algorithm would be impractical for large  $Q$  values. Subsequently, Raghavan *et al.* [Ragh83] presented  $O(n^2)$  algorithms for feasibility as well as realisation. Later [Ragh84] gave an  $O(n)$  algorithm for feasibility. Bhattacharya *et al.* [Bhat88] take a graph theoretical approach and declare crossover free routability if the *Interval Overlap Graph*  $G_o$  is bipartite. Saxena *et al.* [Saxe89] have proposed an  $O(n)$  algorithm ( $O(n \log n)$  on unsorted inputs) for feasibility as well as realisation by constructing a special form of overlap graph  $G_{of}$ , which captures only the first overlap relation. These algorithms will be discussed briefly now in terms of the formulations each used to represent the crossover free routability problem.

### 5.1.1 Formulations of the crossover free routing problem

- *The decision problem KBENDWIRE:* [Ragh84]

The decision problem to be considered is as follows: Given an instance of the SRRP, is there a realisation with at most  $k$  bends (each bend is a right angled turn) in the layout of each wire? For single row geometry,  $k \geq 2$  because each net has to bend at least at the start and end points and T shaped connections for multipin nets are counted as two bends. Further, the number of bends on the layout of any wire depends on the number of crossovers on the net because each crossover results in two additional bends on the wire. Thus  $k = 2 * (C_r + 1)$  where  $C_r$  is the number of crossovers on the net for 2-pin nets. So,  $k$  is always an even number. The special case of KBENDWIRE with  $k = 2$  corresponds to the crossover free realisation, because when  $k = 2$ , each net cannot have any crossover. It is not known whether KBENDWIRE is NP-complete for an arbitrary  $k$  [Ragh84]. However, for the special case of  $k = 2$  polynomial algorithms are possible as described below.

- *Decomposition of netlist into Interlocked sets* : [Ragh83]

Two nets are said to *interlock* if assignment of one net to one street forces the assignment of the other net to the other street in order to achieve a crossover free solution. That is, two interlocking nets cannot run in the same street without causing a crossover. On the other hand, if the assignment of two nets can be made independent of each other then they do not interlock and can run in the same or opposite street without causing crossovers. Examples of interlocking and non-interlocking nets are shown in Figure 5.2.a. An *interlocked set* is a maximal subset of the given netlist such that the assignment of any one net of the set to one street automatically fixes the street assignment of all other nets in the set. Valid interlocking sets are also illustrated in Figure 5.2.b. It is obvious that two nets interlock if they overlap, and they do not interlock if one is completely contained within the other. Thus, every maximally connected component in  $G_w$  is an interlocked set.

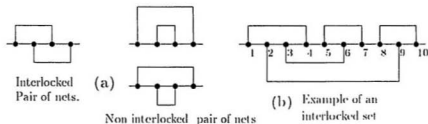


Figure 5.2: Some examples of Interlocking.

An  $O(n^2)$  algorithm is presented by Raghavan *et al.* [Ragh83] for the feasibility problem. The algorithm attempts to partition the netlist into its constituent interlocked sets. Clearly, the netlist has a feasible crossover free solution if and only if such a partition exists. Recently Lloyd [Lloyd89] reported an algorithm of complexity  $O(n \log n)$  to find the interlocking sets based on the observation that one need not construct the entire *interlock graph* in order to locate its connected components (hence, the interlocking

sets). Rather, it is sufficient to construct a spanning forest of the interlock graph. Once the feasibility is established, [Ragh83] gives an  $O(n^2)$  algorithm for the realisation. First the containment relation between the interlocked sets is used to construct a partial order tree. Then the node merging operation by which two sets are merged into a single subassembly is performed using a dynamic programming approach.

- *Planarity testing of blocked version of SRRP* : [Ragh84]

Raghavan *et al.* [Ragh84] gave an elegant algorithm to determine the feasibility of crossover free routing. Consider an SRRP which has  $n$  nodes where the nodes are numbered  $\{1, 2, \dots, n\}$  from left to right augmented with  $n$  new nets  $\{(1, 2), (2, 3), \dots, (n-1, n)\}$  connecting adjacent nodes in the original SRRP. Consider now the graph  $G(V, E)$  of the augmented version, where  $V$  is the set of  $n$  nodes and  $E$  contains all the original nets and the  $n$  new nets.  $G$  is then tested for planarity which can be done in linear time [Hopc79]. If the graph  $G$  is planar, the original SRRP has a crossover free solution. Thus, feasibility of crossover free routing can be established in linear time.

- *Bipartition of the Interval Overlap Graph  $G_o$*  : [Bhat88]

Bhattacharya *et al.* [Bhat88] take a graph theoretic approach to single row routing Problems. Given any SRRP, using the interval model the Overlap Graph  $G_o$  is constructed first. A straight forward procedure to construct  $G_o$  needs  $O(m^2)$  time where  $m$  is the number of intervals (nets of the SRRP) and each interval is compared with every other interval to find the nature of interaction (overlap/containment/no interaction). A better method is to use a scan line approach. If  $G_o$  is bipartite then the SRRP has a crossover free solution. This is because any crossover in SRRP is due to the conflict between two overlapping nets and containment nets do not contribute to crossovers.

- *Construction of first overlap graph  $G_{of}$*  [Saxe89]

Saxena *et al.* [Saxe89] take a slightly different approach from that of [Bhat88].

Instead of constructing the entire Overlap Graph  $G_o$ , they construct a special form of the overlap graph  $G_{of}$ , which captures only the *first overlap* relations. The nets are first ordered based on the left edge. Then using these ordered intervals, the first overlapping interval for each interval is found out and this information is represented as an edge in the  $G_{of}$  between the nodes representing the intervals. The process is then repeated with the intervals ordered based on the right edge. If an edge is required now which does not already exist, then it is added to  $G_{of}$ . After the construction of  $G_{of}$  is completed, the connected components of  $G_{of}$  are found and the spanning forest  $T$  is constructed. If  $T$  is 2-colourable then the original SRRP is bipartite. The realisation can be done then based on the depth of the containments of each containment group. The overall complexity of their algorithm is  $O(n)$  if the input netlist is available in both left edge and right edge sorted order; otherwise, the algorithm requires  $O(n \log n)$  time.

## 5.2 A New approach to the via free routing of Permutation Channels

In this section a simple  $O(n)$  algorithm ( $O(n \log n)$  on unsorted inputs) is presented for finding a via free routing, if any, of a Permutation Channel Routing Problem (PCRP) containing  $n$  nets. This problem has been solved in  $O(n^2)$  time by Even *et al.* [Even72] and in  $O(n \log n)$  time by Supowit [Supo85]. The proposed algorithm finds a via free routing of a PCRP by solving the equivalent problem of finding a crossover free solution of the Permutation Single Row Routing (PSRR) problem where the PSRR is obtained from PCRP using the Single Row Transformation (SRT) approach employing the *abcd* type FT.



### 5.2.1 Permutation Channel / Permutation SRR problems

The Permutation Channel Routing Problem (PCRP) is a special case in which all the nets are 2-pin nets, and for each net one pin is located on the top side of the channel and the other on the bottom side of the channel; *i.e.*, there are no *local* nets. A PCRP can be described as the matching diagram of a Permutation Function  $\pi$ . An example of a PCRP corresponding to the permutation  $\pi = [2, 5, 4, 1, 3]$  is illustrated in Figure 5.3. The PCRP can also be modeled as a Permutation Graph where the nodes represent the nets of the channel and the intersection of the nets defines the edges. In fact the Permutation Graph is a special case of the more general *circle graph* [Hsu86] model of a channel.

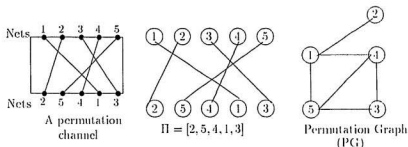


Figure 5.3: A Permutation Channel Routing Problem (PCRP)

In Chapter 4 it was explained that while many FT-BT pairs are possible, only simple pairs are practical and the *abcd* FT was chosen as the candidate for the development of a channel router based on the SRT approach. In addition to the simplicity, the *abcd* FT results in a number of comparable properties between the channel and its corresponding SRR. For example, the *circle graph* of the channel is identical to the *Interval Overlap graph*  $G_o$  [Bhat88] of the SRR and crossovers in SRR result in vias in the channel solution. However, the number of vias in the final channel need not be the same as the number of crossovers in the SRR solution due to *crossover collapsing* that may take place.

The SRR problem which is obtained from a Permutation channel using *abcd* FT is called PSRR. Note that the PCRPP will have a via free solution only if the PSRR has a crossover free solution. A bipartite PSRR is called PBSRR. An example of a PCRPP which translates into a PBSRR under *abcd* FT is illustrated in Figure 5.4. Now some interesting properties of a PSRR are identified. First of all, the PSRR is a *Single group* [Du86] SRR because all the nets cut across a common vertical line YY (Figure 5.4), which means that any two nets of a PSRR should intersect (*overlap or contain*). No two nets are isolated and thus the overall interval graph  $G_I$  is a clique. The *node cut numbers* uniformly increase and then decrease, as the nodes are scanned from left to right. Finally, the total length of all nets in a PSRR of  $m$  nets is  $m^2$ .

## 5.2.2 Previous work on via free routing of a PCRPP

Since vias in a channel solution have many disadvantages, minimising the via count is important. For via free routing of a channel, each net in the channel must be routed totally in one layer. Assuming a 2-layer model, this amounts to the task of partitioning the set of nets into 2 groups such that no two members of a group intersect, so that each group can be laid out on a single layer. Thus, channel via-free routing can be formulated as the graph theoretic problem of *bipartitioning* the circle graph  $C_g$  of the channel. Consequently, any cycle of odd length in  $C_g$  would make it non-bipartite and then the channel has no via free solution.

The graph theoretic term *minimum node colouring* of a graph  $G = (V, E)$ , refers to the minimum number of colours that is needed to colour the nodes of  $G$  such that no adjacent nodes have the same colour. This problem is also referred to as the *minimal chromatic decomposition*. If a graph is bipartite, then it is 2-colourable and the set of nodes of  $G$  can be decomposed into two independent sets. The problem of minimum colouring of circle graphs has been shown to be

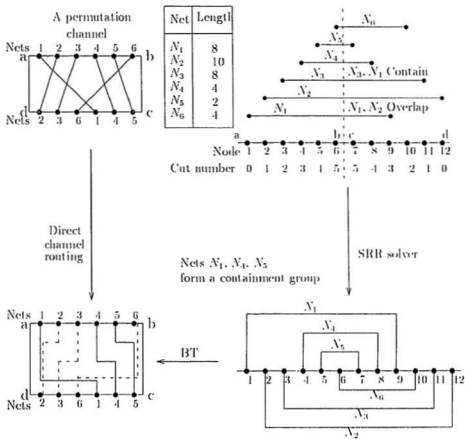


Figure 5.4: Permutation Channel routing by SRT approach.

NP-complete [Gare79]. However, in the case of a Permutation Channel the circle graph reduces to the Permutation Graph which can be coloured in polynomial time. Even *et al.* [Even72] have given a  $O(n^2)$  algorithm to find a minimal chromatic decomposition of a permutation graph. The algorithm considers the nets one by one and incrementally builds up the *independent sets*. If only two sets result at the end of the process, then the PCRP is bipartite. Even *et al.* have also indicated that if one reads the  $\pi$  vector from left to right and identifies the *increasing subsequences* (which need not necessarily contain adjacent elements of  $\pi$ ), each such sequence corresponds to an independent set. For example, the problem in Figure 5.3 has a  $\pi = [2,5,4,1,3]$ , the increasing subsequences are  $[2,5]$ ,  $[4]$  and  $[1,3]$  and the example is non-bipartite.

Supowit [Supo85] gives an  $O(n \log n)$  algorithm for minimum colouring of a PCRP by solving an equivalent problem, namely, partitioning a set of points in a plane into a minimum number of *ascending chains*. An example of this approach is shown in Figure 5.5. The points are first sorted based on their  $x$ -coordinate. Then the chains are built using the following rules: The next new point is added to that chain whose highest  $y$ -coordinate is less than the  $y$ -coordinate of the new point. When there is more than one choice, that chain whose highest  $y$ -coordinate is closer to the  $y$ -coordinate of the new point is chosen. When there is no choice, a new chain is started with the new point as its first member. This scheme results in the decomposition of the given points into an optimal number of chains.

It is worth noting that the greedy approach of finding a 2-independent set as two sequential 1-independent set problems may not always lead to an optimal solution. Figure 5.5, finding the Maximum Independent Set (MIS) as the longest up-sequence [Golu80] in  $\pi$  namely  $\{1,2,5,6\}$ , leaves behind nets 3 and 4 which intersect and should form two more separate sets. Thus, the chromatic decomposition is  $\{1,2,5,6\}$ ,  $\{3\}$  and  $\{4\}$  which is clearly a non-bipartite solution. However, the problem has a bipartite solution where the sets are  $\{1,2,3\}$  and  $\{4,5,6\}$ .

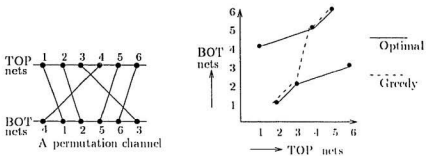


Figure 5.5: Minimum number of *ascending chains* approach

### 5.2.3 Some new results on via free routing of a PCR<sup>P</sup>

In this section, the problem of via free routing of a PCR<sup>P</sup> is solved by finding the crossover free solution of the corresponding PSRR. Consider a PCR<sup>P</sup> specified by the netlist  $N = \{N_1, N_2, \dots, N_m\}$ . Every net in  $N$  is a 2-pin net and connects one pin of the TOP row of the channel to one pin of the BOTTOM row of the channel. Each of these *permutation nets* can be classified as a *left net*, a *right net* or a *straight net*. Consider a net  $N_i$ <sup>1</sup> which connects the  $i^{\text{th}}$  terminal of the top side (from left end) to the  $j^{\text{th}}$  terminal of the bottom side. Then  $N_i$  is a *left net* if  $j < i$ . That is, the net is directed towards left. If  $j > i$  then the net is a *right net*, and if  $i = j$  then the net is a *straight net*. In Figure 5.4  $N_1$ ,  $N_4$  and  $N_5$  are right nets,  $N_2$ ,  $N_3$  and  $N_6$  are left nets and there are no straight nets. Some results regarding the left, right and straight nets of a PCR<sup>P</sup> follow. In the discussions following, note that there are no unconnected terminals in the channel.

<sup>1</sup>In the PCR<sup>P</sup> model, the TOP side of the channel has the net numbers  $1, 2, 3, \dots, m$ , in the increasing order and the BOTTOM side net numbers are given by the vector  $\pi$ , as seen in the examples illustrated so far. If however, in a general case, if the TOP side net numbers are not in increasing order, then sorting them and changing the BOTTOM  $\pi$  vector correspondingly, so that we have a PCR<sup>P</sup> conforming to our model, can be done in an additional  $(m \log m)$  time. Thus, a net which start at the  $k^{\text{th}}$  TOP node from the left, will be labelled net  $N_k$ .

**Lemma 1.** Every left net of the PCRPP is cut by at least one right net. Similarly every right net of the PCRPP is cut by at least one left net.

*Proof :* Consider a right net  $N_i$  connecting the  $i^{th}$  node of top side of the channel and  $(i + k)^{th}$  node of the bottom side of the channel, as shown in Figure 5.6.a, where  $1 \leq k \leq (m - 1)$  where  $m$  is the total number of nets. This net divides the top side into two parts A and B and the bottom side into two parts C and D. There are  $i - 1$  nodes in A,  $m - i$  nodes in B,  $i$  nets in C and  $m - i - 1$  nets in D. In a PCRPP, every node on the top side must be connected to some node in the bottom side. So considering the parts A and C, a maximum of  $i - 1$  nets could be totally to the left of the net  $N_i$  which means that there will be a terminal unconnected in C which must connect with a terminal in B, constituting a left net that will cut across the right net  $N_i$ . Similarly, every left net will be cut by at least one right net.

**Corollary 1:** If two left (right) nets cross in a PCRPP (overlap in the corresponding PSRR), then the conflict cannot be resolved without vias (crossovers).

Since every left net cuts at least one right net and vice versa, right and left nets should be assigned to different layers for a via free solution of a PCRPP. In the same way, in the PSRR to get a crossover free solution, all the left nets should be routed in one (say upper) street and all right nets should be routed in the other (lower) street. Thus, if two left nets cross in the channel (overlap in the PSRR), the other layer cannot be used for resolving the conflict because it is already preassigned for right nets.

**Lemma 2.** Every straight net of the PCRPP is either cut by no other net or cut by at least one left net and at least one right net. In fact every straight net of a PCRPP is cut by an even number (possibly zero) of left and right nets.

*Proof :* Consider a straight net  $N_k$  which connects the  $k^{th}$  nodes in the top and

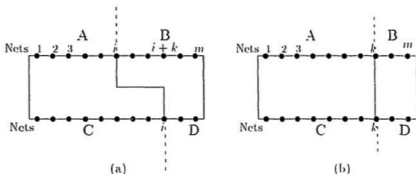


Figure 5.6: Properties of the nets of a PCR

bottom, as shown in Figure 5.6.b, where  $1 \leq k \leq m$ . The net  $N_k$  divides the top into two parts A and B and the bottom side into two parts C and D. There are  $k - 1$  nodes in A and C each and  $m - k$  nodes in B and D each. Considering the parts A and C, all the  $k - 1$  nets could be totally to the left of  $N_k$  and considering B and D, all the  $k - 1$  nets could be totally to the right of  $N_k$  which means  $N_k$  is not cut by any net. Now assume that there is a right net which connects a node from A to a node in D, then this will cut across  $N_k$ . This leaves  $k - 2$  nodes in A and  $k - 1$  nodes in C unconnected and a maximum of  $k - 2$  nets can then be totally to the left of  $N_k$  which means there will be at least one unconnected node in C which must be connected to a node in B, which will constitute a left net cutting across the straight net  $N_k$ . Similarly, if a straight net is cut by a left net it will be cut by a right net as well. The straight nets which are not cut by any net are called *straight non-cut nets* and those which are cut by at least one left and one right net are called *straight cut nets*.

*Corollary 2* : A straight non-cut net partitions the problem into two independent subproblems while a straight cut net leads to non-bipartition.

**Theorem 1.** A PCR can have a via free solution if and only if no two left nets intersect, no two right nets intersect and there are no straight cut nets. Proof

follows from Corollary 2.

**Lemma 4 :** A PSRR is crossover free (*i.e.*, it is a PBSRR), if and only if every two left (right) nets have containment relation. That is, all the left (right) nets should form a *containment group*.

*Proof :* Every two nets of a PSRR should *intersect*, where the intersection could be either *overlap* or *containment*. Now considering only the left nets, no two of them can overlap in a crossover free solution. Thus, every two left nets should contain which means all left (right) nets form a containment group. An example of a containment group is shown in Figure 5.4.

*Corollary 3 :* A PSRR is non-bipartite if any three nets have the same length.

## 5.2.4 Description of the new algorithm

**Input :** A PCRP specified using the permutation function  $\pi$ .

**Output:** A via-free solution, if any, for the PCRP.

**Step 1:** *Convert the PCRP into a PSRR problem*

The number of elements in  $\pi$  gives the number of nets  $m$ . Read the  $\pi$  vector from left to right. If  $\pi(i) = k$ , ( $1 \leq i \leq m$ ), then the length of net  $N_k$  in the PSRR model is  $(2 * m) + 1 - i - k$ . This is illustrated in Figure 5.4. This step takes  $O(m)$  time and now the length of all nets of the PSRR is known.

**Step 2:** *Identify the Left, Right and Straight nets in the PSRR model*

Let  $L_i$  be the length of net  $N_i$ , where  $N_i$  starts at the  $i^{th}$  node ( $1 \leq i \leq m$ ) of the PSRR. This would be true if the net numbering is done based on their starting node order. However, if the nets are not numbered using such an order, it can be done in  $O(m \log m)$  time by a sorting and renaming process, so that the net



starting at node  $i$  can be called net  $N_i$ . Then the following holds true.

If  $L_i > 2 * (m - i) + 1$  then  $N_i$  is a Left Net.

If  $L_i < 2 * (m - i) + 1$  then  $N_i$  is a Right Net.

If  $L_i = 2 * (m - i) + 1$  then  $N_i$  is a Straight Net.

In the case of straight net its type should be identified as well, *i.e.*, cut or non-cut. To do this, as the nets  $\{N_1, N_2, \dots, N_m\}$  of the PSRR are scanned from left to right, keep track of the length of the shortest net using a variable, say  $L_{min}$ . If  $N_k$  is a straight net and if  $L_k < L_{min}$  then  $N_k$  is a straight non-cut net, else it is a straight cut net and bipartition fails. The overall time complexity of this step is  $O(m)$ .

**Step 3:** *Find if the left (right) nets make a containment group*

In a containment group, the shortest net will run closest to the node axis of the PSRR and nets of increasing length will run on the tracks successively further from the node axis. Let  $N_i$  and  $N_j$ , ( $j > i$ ), be two adjacent left (right) nets, as the nets  $N_1, N_2, \dots, N_m$  of the PSRR are scanned. Then, if the relation  $(j - i) < (L_i - L_j)$  is true then  $L_i$  contains  $L_j$ . Now test this relation for every pair of adjacent left (right) nets. A maximum of  $m - 1$  comparisons would be needed and thus the complexity of this step is  $O(m)$ . Failure of this test means that two left(right) nets cross, and no bipartition is possible. If *feasibility* is ensured, continue for *realisation*.

**Step 4:** *SRR realisation*

At this point all the left (right) nets make a containment group. All the left nets can be routed in one street (say upper) and all the straight nets can be routed in the other (lower) street without conflicts. However, there may be some straight non-cut nets as well and the question arises as how to route them. It is easily seen that a straight non-cut net divides the PCRP into two independent subproblems and hence every straight non-cut net can be routed in the upper or lower street

without cutting other nets. However, allotting the straight non-cut nets to the less congested street will help in achieving nearly equal congestion of either street. A random left or right grouping would still produce a crossover free solution, which may not be the least congested among the many possible crossover free solutions. Thus, at the end of this step, every net in the PCR<sub>P</sub> has been labelled as a Left or a Right net. This colouring will be referred to as **LR-colouring** in future.

From the structure of the containment group it is obvious that the net of shortest length in the group (which will also have the highest left node) should be closest to the node axis, *i.e.*, that net should occupy the track  $T_{\pm 1}$ . Since no two nets of a PSRR can occupy the same track (because any two nets of PSRR either overlap or contain), the other nets of the group should be placed in tracks away from the node axis. This ordering of nets which is based on the decreasing order of their left (starting) node will be referred to as **box ordering** in future. Box ordering results in the order of decreasing length of nets within a containment group. Any PBSRR can be easily routed using the **box procedure** which refers to routing an SRR problem using *LR-colouring* and *Box ordering*.

**Step 5: Convert the PBSRR layout to PCR<sub>P</sub> layout**

The last step in the process is to fold the SRR layout into the channel layout. All the upper(lower) street nets of the PSRR are mapped to layer 1(2) of the channel, as illustrated in the example shown in Figure 5.4.

**Theorem 3 :** A PCR<sub>P</sub> can be declared bipartite or not in  $O(m)$  time  $\{O(m \log m)$  time if the netlist is unsorted $\}$ , where  $m$  is the total number of nets. If the problem is bipartite, then it can be routed as well in  $O(m)$  time. Proof of the theorem follows immediately from the above discussions.

### 5.2.5 Extensions to the proposed algorithm

The simplicity of the *box procedure* approach to route the PSRR problem provided the encouragement to explore its possible extensions to some other problems of interest within the frame work of SRT based channel routing. Three extensions that were examined are outlined below.

- The MBSRR (Mixed bipartite SRR) problem.

This was motivated by the fact that polynomial time algorithms had been proposed [Ragh84, Saxe89] for the crossover free routing of any SRR problem. Raghavan *et al.* [Ragh84] gave a linear time algorithm (based on planarity checking) for determining the feasibility of crossover free routing and an  $O(N^2)$  algorithm based on dynamic programming for the subsequent realisation. Saxena *et al.* [Saxe89] described  $O(n \log n)$  algorithms for feasibility and realisation, using a different formulation. Thus, it is clear that the complexity of *crossover free routing* for any SRRP is only  $O(n \log n)$  and it may be possible to find an  $O(n \log n)$  algorithm for the MBSRR routing based on another suitable formulation. Since the time complexity of the box procedure is  $O(m \log m)$  on unsorted input netlist containing  $m$  nets, it seemed to be a good candidate for MBSRR routing as well, but the extension is not straightforward. The main reason is that the MSRR problem is a more general problem than the PSRR problem and hence the special properties of the PSRR problem (that were used in the box procedure) cannot be carried over easily to the MSRR problem. The problems faced in extending the box procedure to MBSRR are discussed in more detail under the next section on MBSRR routing.

- The PNSRR (Permutation non bipartite SRR) problem.

The next extension that was attempted was to apply the box procedure to route a PNSRR with minimum crossovers. It is clear that minimising the

crossovers in the SRR solution would decrease the number of crossovers to be aligned during the BT step, which would make the BT step easier. Further, fewer crossovers in the SRR solution usually lead to a reduction of vias in the final channel. Although there is no one-to-one translation of crossovers into vias in the general case (for example, due to *crossover collapsing*), the number of crossovers act as an upper bound on the number of vias, when there are no difficult crossovers. Thus, it is worth attempting to minimise the number of crossovers in the SRR solution. As will be explained in a subsequent section, *L-L*, *R-R* and *L-R-S* overlaps in the interval model (where *L* refers to left nets, *R* refers to right nets and *S* refers to straight nets), translate into an essential via in the final channel, if no detouring is allowed. Experimentation with different examples has shown that the box procedure usually results in a *minimum*<sup>2</sup> number of crossovers for a PNSRR. Both the problems of PNSRR with no straight nets and PNSRR with straight nets were considered and the details will be discussed in the next chapter on non-bipartite routing.

- Topological Via minimisation (TVM) of Permutation Channel.

Topological via minimisation (TVM) was proposed by Hsu [Hsu83], where the via minimisation is an integral part of routing. TVM has been shown [Sadw84, Sarr89] to be NP-hard even for the 2 layer channel routing case with two terminal nets. Thus, heuristic algorithms are justified and some have been reported [Hsu83, Chan87]. However, for the restricted case of the permutation channel (also called split channel) routing problem (PCRP), Sarrafzadeh *et al.* have proposed a  $O(n^2 \log n)$  algorithm [Sar89] for TVM. Recently, Rim *et al.* have shown [Rim89] that TVM of  $k$  ( $k \geq 2$ ) layer permutation channels with 2-pins is solvable in  $O(k * n^2)$  time. Thus, TVM of 2 layer permutation channels is solvable in polynomial time.

---

<sup>2</sup>It is to be recalled that the term *minimum* crossover of the SRR solution within the framework of SRT based routing, does not necessarily mean the *absolute minimum* count of the crossovers.

The  $O(n^2 \log n)$  algorithm proposed by Sarrafzadeh *et al.* for the TVM of 2-layer permutation channels, consists of two steps, namely, *solid net identification* which finds a maximum bipartite subset of the original netlist in  $O(n^2 \log n)$  time and then the *residual net routing* using the theorem [Sadw84] that every residual net can be routed with at most one via by proper pushing around of other nets. This approach assumes detouring (which is meandering beyond the end points of the nets and requires external doglegs to connect the segments of the same net at different levels) is allowed. Of course, in the SRT based router environment, this is a concern to be addressed during the BT step. Thus, in addition to the box procedure based routing, a suitable mechanism to allow detours in the final channel is essential in the BT step, making the BT more complicated. Further, a suitable *crossoner collapsing* technique is needed in the BT step to map the crossovers onto essential vias. With these enhancements TVM of 2-layer PCR<sup>2</sup> is possible using the box procedure.

### 5.3 MBSRR routing

In this section the problem of routing a MBSRR is addressed. The MSRR problem should be declared as bipartite (MBSRR) or not; if bipartite, a crossover free realisation should be found. The MSRR is a *mixed SHR* problem in the sense that in addition to *permutation nets*  $\{\text{Left}(L), \text{Right}(R) \text{ and } \text{Straight}(S)\}$ , there can be one or more *local nets*. A net is *local* if all its terminals lie on the same side of the channel. Without loss of generality, only 2-pin nets are assumed because each multipin net can always be decomposed into a number of 2-pin nets and then collapsed. Thus, the main difference between PBSRR and MBSRR is that MBSRR allows local nets.

### 5.3.1 MBSRR routing using box procedure extension

Recall that the box procedure for routing a PBSRR involves two steps, namely, LR-colouring and box-ordering. In the LR-colouring process, every Left net is labelled  $L$ , every Right net is labelled  $R$  and the straight nets are labelled as  $L$  or  $R$ . While this is possible in the MBSRR case also, the problem arises as how to colour the local nets. Also two important properties<sup>3</sup> of the PSRR problem are no longer true for the MSRR, due to the presence of the local nets. Consider the example shown in Figure 5.7.a, where  $L$ ,  $R$  and  $S$  nets have to be routed on the same layer to achieve a crossover free solution. Thus, the philosophy of keeping  $L$  and  $R$  nets on opposite streets is not valid for MBSRR routing. Some interesting properties of the MSRR problem follow.

- For every local net on the top side of the channel there is a local net on the bottom side of the channel. Thus, the total number of local nets is always even.
- A straight net is either uncut or cut by an even number of nets.
- An SRR problem in which three nets overlap with each other resulting in a 3-cycle in the overlap graph, cannot have a crossover free solution. Thus, all nets which overlap a local net should not overlap with each other to get a crossover free realisation. This helps to assign a local net to one street and all the nets which overlap with it to the opposite street. However, such an assignment is not possible when there is *direct* or *indirect coupling* between local nets, as shown in Figure 5.7.b. Thus, local nets cannot be coloured in isolation but should be coloured in connected groups. This principle is used in a later section on MBSRR using interlocked sets.

---

<sup>3</sup>1. Every left net is cut by at least one right net and vice versa.

2. Every straight net is cut by no net or cut by both a left and a right net.

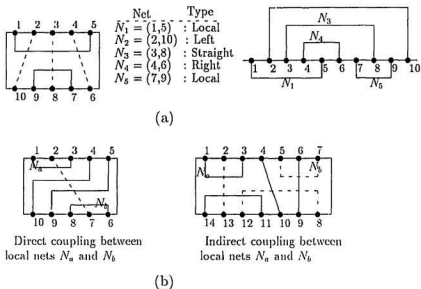


Figure 5.7: Example of MBSRR routing

Another important difference between PBSRR and MBSRR becomes clear by analysing the legal structure of nets in the same street of a crossover free solution. First of all, for a Bipartite SRR problem, any two nets on the same street should either be non-intersecting or containing. *i.e.*, no two nets in the same street can overlap. Nets on the same street can form simple or nested containments. In a simple containment, which will also be referred to as a *containment group*, any two nets have a containment relation. However, in nested containments, any two nets can contain or remain non-intersect. It is known that the PSRR problem is a Single Group SRR problem (*i.e.*, all nets cut across a common axis, which means no net can start after another net has ended) and thus any two nets of PSRR should intersect *i.e.*, should overlap or contain. Thus, for the PBSRR case, all the nets which can be routed on the same street should contain and form a single *containment group* as shown in Figure 5.8.a. This single containment group is no longer true for MBSRR which can have nested containment groups or multiple isolated

containment groups, as illustrated in Figure 5.8.b and Figure 5.8.c. Thus, neither LR-colouring nor Box ordering holds true for MBSRR and hence it is no surprise that the box procedure fails to route MBSRR. This led to solving the MBSRR using other formulations as described below.

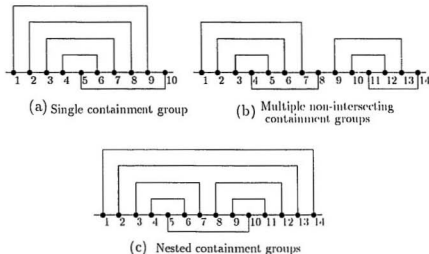


Figure 5.8: Legal structure of nets in the same street of a bipartite SRR

### 5.3.2 MBSRR routing using interlocked set approach

An *interlocked set* of nets was previously defined to be a set of nets such that assignment of any one of the nets to one layer automatically forces the rest of the nets in that set to the upper/lower street to get a crossover free realisation. One approach to route a MBSRR is to decompose the netlist into its constituent interlocked sets using a previously described algorithm [Lloy89], with a time complexity of  $O(n \log n)$ . If the attempt is successful, then there is a feasible crossover free solution. If the Overlap Graph  $G_o$  is connected then the MBSRR will contain only one interlocked set which includes all the nets. If  $G_o$  is unconnected then there will



be one interlocked set for each of the connected components of  $G_o$ . Once the feasibility is determined, then each of the interlocked sets can be routed independently of others. That is, 2-colour each of the interlocked sets (which can be done in linear time, by say a depth first search) and assign them to tracks one after another. Of course, if one interlocked set is totally contained within the gap of another, then the smaller interlocked set will have to be routed before the bigger. Examples with single as well as multi-interlocked sets are illustrated in Figure 5.9.

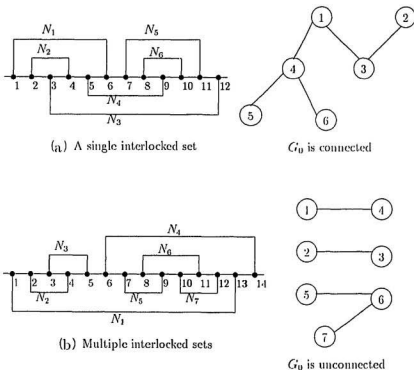


Figure 5.9: Examples of single and multi-interlocked MBSRR

Based on the discussion of the structure and properties of MBSRR so far, it is clear that to find a crossover free realisation of an MBSRR two pieces of information are needed about each net. They are the *strict information* which

tells whether the net runs in the upper (+) street or the lower (-) street (this  $\pm$  information will also be referred to as colouring or 2-colouring information) and *level* or *track number* of each net within its street. It can be seen that in the absence of crossovers the entire stretch of each net lies in only one level. Thus, with the *street* and *level* information for each net, a crossover free realisation can be uniquely specified. The *street* for each net can be decided by constructing the overlap graph  $G_o$  of the SRR problem and then 2-colouring it. If  $G_o$  is unconnected, many 2-coloured solutions are possible, each corresponding to a crossover free solution, with one of them being chosen. The colouring information alone is not enough to realise the MBSRR layout, because in each containment group (either single or nested), the inner nets should be routed before the outer nets which calls for an ordering. This additional ordering should essentially capture the containment relationship between nets. It is now proposed that *Tarng's ordering* can be used for the purpose of specifying the order in which the nets should be routed, once the street assignment of each net is over. A new algorithm called *TARNG-MOD*, which is a modified version of the original algorithm of Tarng *et al.* [Tarn84] is described below.

## 5.4 Modified Tarng's algorithm

This section describes modifications to Tarng's algorithm [Tarn84], to obtain a crossover free solution for a given SRRP. Recall that Tarng's algorithm does not necessarily produce a crossover free solution of a SRRP, even though one exists, because Tarng's algorithm was designed with the goal of track optimisation and not crossover minimisation. An example where Tarng's algorithm fails to produce a crossover free solution although one exists is shown in Figure 5.10. The solu-

tion in (a) is a possible<sup>4</sup> solution by Tarng's algorithm and the solution in (b) is a crossover free solution, which is not produced by Tarng's algorithm. In this particular example, both the crossover free and crossovering solutions have the same street congestion  $Q = 2$ , which need not be true in general. In fact, in many cases the goals of crossover minimisation and track optimisation are contradictory, in the sense that a solution with fewer crossovers usually needs more tracks and in such cases priority is to be given to a crossover free solution. Thus, for SRT based routers solution (b) is preferred to solution (a). This leads to the question as to whether there is a way by which Tarng's algorithm can be modified so that it produces a crossover free solution, if one exists, although it may be more congested than the theoretical minimum congestion of  $Q_0$ .

#### 5.4.1 Tarng's algorithm revisited

Before outlining such a modification, it is useful to recapitulate the salient points of Tarng's algorithm detailed in Chapter 3.

- The goal of Tarng's algorithm is *track optimisation*. The algorithm evolved from the heuristic that nets of high cut numbers should be kept close to the axis. That is, as one proceeds away from the axis on either side, nets should appear with decreasing cut numbers. To achieve this the 3-step algorithm was devised, the steps being *Zone allotment*, *Net ordering* and *Track assign*.
- In *Zone Allotment* nets are grouped into zones based on their *internal cut numbers*. In the second step, nets within each zone are further ordered based on their *residual cut numbers*. At the end of this step, all the nets are ordered and this order is referred to as *Tarng's ordering*. Within a zone it is possible

---

<sup>4</sup>It should be recalled for a given SRFP, multiple solutions are possible using Tarng's algorithm depending upon the choice made for the property 1 and 2 discussed in Chapter 3.

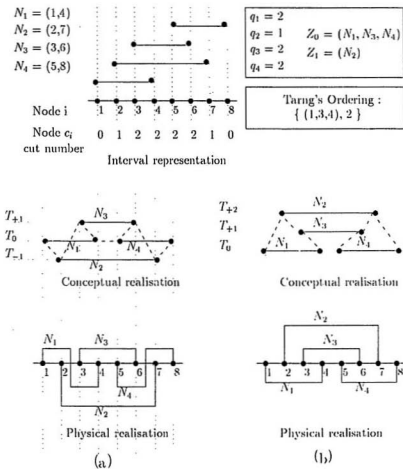


Figure 5.10: Targ's algorithm's failure in crossover free routing.

that two or more nets have the same residual cut number; then their ordering is arbitrary. This will be called *property 1*. For example, in Figure 5.11 the nets  $N_1, N_3$  and  $N_4$  of the first zone have the same residual cut number and hence their ordering is arbitrary. Such a group of nets is enclosed in round brackets. In the last step of *track-assign*, nets are taken as per Tarnig's ordering and assigned to *pseudo tracks* to form a *conceptual realisation*, using two principles. First, choose the track for placing a net such that the absolute value of the track index is minimised. This means the choice of  $T_{+j}$  or  $T_{-j}$  does not matter. This will be referred to as *property 2*. Only the outermost tracks used so far can be shared when a new zone is started. This is needed to satisfy the requirement of having decreasing cut numbers as one proceeds away from the axis.

Experimentation with different examples has shown some interesting results about the effect of property 1 and property 2 of Tarnig's algorithm. For a single group SRR, property 1 and property 2 do not matter for *street optimality* but they do matter for *crossover optimality*. For a multigroup SRR, these properties affect both the street optimality and the *crossover optimality*. Since these two properties matter for the crossover optimality of both single and multigroup SRRs, it seems that something needs to be changed about these two properties to obtain optimal crossovers.

More experimentation led to the understanding that the philosophy of *minimising the absolute value of track index* while assigning nets to pseudo tracks is the crux of the problem. This philosophy was based on the requirement to keep the upper and lower street congestions nearly equal which may contradict crossover optimality. This is evident from the example shown in Fig. 5.10 where the *track index minimisation* strategy forces  $N_2$  to track  $T_{-1}$  but it must be put at  $T_{+2}$  to get the crossover free solution, which needs containment of  $N_2$  and  $N_3$  to be in

the upper street. Thus, it is clear that the track index minimisation has to be dispensed with in favour of another philosophy which is geared towards allowing such containments.

### 5.4.2 Details of the new algorithm TARNG-MOD

In this section, it is proposed that the 2-colouring order obtained by bipartition of the overlap graph  $G_o$  can be used in place of Tarnig's track index minimisation philosophy. Experimentation has shown that routing the nets based on Tarnig's ordering with the 2-colouring information deciding the street assignment leads to a crossover free solution, if one exists. Tarnig's ordering step is still retained because, as seen earlier, although the conceptual realisation depends on Tarnig's ordering as well on the track assign philosophy, it is the latter step which is the main reason for getting the crossovers. Another important advantage of this approach is that for the bipartite examples, the steps of conceptual realisation and topological mapping (by drawing the reference line) can be bypassed altogether and the physical realisation can be directly obtained. This is because for the bipartite case the entire net occupies a single track in both the conceptual and physical realisation and this obviates the systematic mapping procedure from conceptual to physical realisation. The nets can be directly assigned to the real tracks  $T_{+1}, T_{+2} \dots$  on the upper street and to the real tracks  $T_{-1}, T_{-2} \dots$  on the lower street. The overall algorithm is described below and is also illustrated through the MBSRR example in Fig. 5.11.

- **Step 1:** Given an SRRP, draw its overlap graph,  $G_o$ . Attempt bipartition of this graph. If not successful, the problem has no crossover free solution. (In the next chapter such cases will be considered). If  $G_o$  is bipartite then colour each net as + or - to denote the upper and lower street assignments respectively.

- **Step 2:** The steps of *Zone allotment* and *Net ordering* of Tarnig's algorithm are used to obtain *Tarnig's ordering*.
- **Step 3:** Now take the nets one by one based on Tarnig's ordering. Assign them to streets based on the + or - colouring information. Here the assignment is done directly onto the first available real track of the physical realisation which has tracks  $T_{+1}, T_{+2} \dots$  in the upper street and tracks  $T_{-1}, T_{-2} \dots$  in the lower street.

The proposed TARNG-MOD algorithm superimposes the Tarnig's ordering on the 2-colouring information to obtain a crossover free solution. But there are alternative ways by which a crossover free realisation could have been obtained once the 2-colouring information is available. That is, the containment ordering information could have been obtained in other ways instead of using Tarnig's ordering. One way, for example, is to construct the containment graph  $G_c$ , graph which captures only the containment and not the overlap relation between the intervals, of the nets in the same street. This graph would be a tree or a collection of trees, where each tree corresponds to a single or nested containment. Now the track number of each net in a containment group can be interpreted as the depth of the corresponding node in the tree. Thus, the nets represented by each tree can be routed from leaf to root with the colouring information deciding the street of all nets in the tree.

While concluding this chapter a few remarks are in order. Although the box procedure is simple it works only for the PBSRR problem. For the MBSRR problem a new algorithm called TARNG-MOD was proposed and other approaches were also outlined. Since the SRR solver in a SRT based router should be capable of handling both MBSRR and PBSRR problems, the *box procedure* is not suitable and TARNG-MOD is chosen to route any bipartite SRR problem.

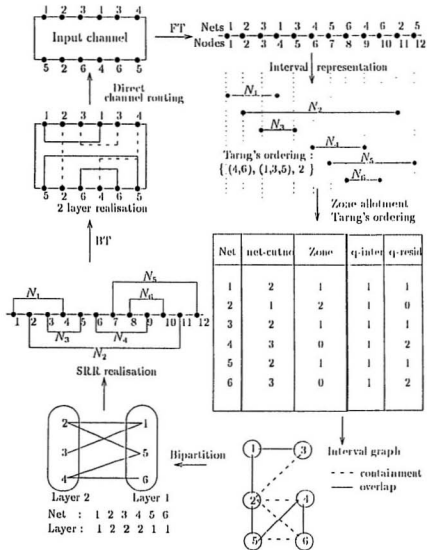


Figure 5.11: TARNG-MOD algorithm based routing of a MISRR



## Chapter 6

# Non-Bipartite Single Row Routing

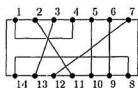
In this chapter, routing of the non-bipartite single row routing problems, namely, the PNSRR (Permutation non-bipartite SRR) and the MNSRR (Mixed non-bipartite SRR) problems, are considered. Given a PNSRR/MNSRR problem, the goal is to find a realisation with *minimum* crossovers. This chapter starts by proposing some new terminology applicable to the minimum crossover routing of non-bipartite SRR problems. A two level approach is pursued to explore this new problem of *minimum crossover routing*. In the first level, minimum crossover routing is understood better by considering in detail the closely related problem of *Topological via minimisation* (TVM) of channels and drawing useful comparisons between the TVM and minimum crossover routing problems. In this context, an efficient heuristic algorithm for the *minimum node deletion* problem is proposed. In the second level, minimum crossover routing is analysed as a problem in its own right, and new algorithms as well as modifications to existing algorithms are proposed for minimum crossover routing of the PNSRR and the MNSRR problems. Finally, to route the MNSRR problem, which is the most general and hence the most difficult to route

non-bipartite SRR problem, it is proposed that the concept of *grouping* [Dul87] be used.

## 6.1 Minimum crossover routing

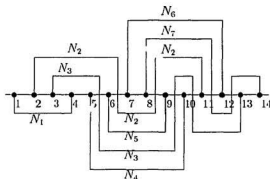
The term *minimum crossover* in an SRT environment does not necessarily mean the absolute crossover count minimisation pursued in an isolated SRR environment because the absolute crossover count does not reflect difficulties posed by the crossovers while carrying out the Backward Transformation. That is, two SRR solutions may contain different numbers of crossovers and yet could be translated, by proper crossover handling techniques in the Backward Transformation step, into channels of an equal number of vias and hence should be considered equal in terms of crossover count. Figure 6.1 shows two SRR realisations of a given non-bipartite SRR problem, which are quite different in their absolute crossover count, but both of which could result in the same number of vias in the channel. In the realisation with seven crossovers, net  $N_3$  has three pseudo crossovers all of which can be collapsed into one via in the channel. Similar collapsing is possible for the two pseudo crossovers on net  $N_2$ . Further, two real crossovers occurring in adjacent intervals can also be collapsed into one via. For example, the adjacent real crossovers on net  $N_7$  of Figure 6.1 can be collapsed as a via on net  $N_6$ . This phenomenon of crossover collapsing has a strong impact on the definition of minimum crossovers in an SRT environment.

Crossover minimisation in an absolute sense does not differentiate between pseudo and real crossovers as well as the collapsible and the non-collapsible. Tsukiyama *et al.* [Tsuk80] have shown that finding a realisation that minimises the total number of crossovers is NP-complete. Thus absolute crossover minimisation is intractable. Bhattacharya *et al.* [Bhat88] have used a graph theoretic approach

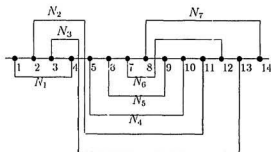


$N_1 = (1,4)$   
 $N_2 = (2,11)$   
 $N_3 = (3,13)$   
 $N_4 = (5,10)$   
 $N_5 = (6,9)$   
 $N_6 = (7,12)$   
 $N_7 = (8,14)$

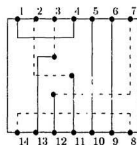
Channel problem



A solution with 7 crossovers



A solution with 3 crossovers



A solution with 3 vias

Figure 6.1: Realisation with different crossovers leading to same via count

and have shown that given an SRR problem, any of its physical realisation needs at least  $C_o$  crossovers, where  $C_o$  is the difference between the total number of nets and the number of nets in a maximum bipartite subgraph of the overlap graph  $G_o$ . Only for special cases of  $G_o$  they have reported the exact value of  $C_o$ . For example, if  $G_o$  is a simple path or even cycle  $C_o$  is zero and if  $G_o$  is an odd cycle then  $C_o$  is one. The general problem of finding a maximum bipartite subgraph of a graph is NP-complete [Gare79] and thus specifying a value for  $C_o$  is not feasible.

For the SRT based environment, although it is desirable to have the minimum number of crossovers  $C_o$ , for a given SRR problem, it is not essential. This follows from the fact that the number of pseudo crossovers in the SRR solution does not really matter since they collapse into only one via in the channel, with suitable crossover handling techniques in the Backward Transformation. Since the entire stretch of a straight net is pseudo any number of crossovers does not matter, whereas the entire stretch of local nets is real and hence crossovers do matter. Left and right nets have both pseudo and real stretches. Since each real crossover needs handling (except when two adjacent crossovers can be collapsed into one via) preference should be given to placing the crossovers on pseudo stretches.

All the SRR algorithms proposed so far use track congestion minimisation as their primary goal. Du *et al.* [Du87] have considered the somewhat related problem of SRR routing with crossover bound, which only restricts the maximum number of crossovers between any two nodes and exploits this criterion to prune unacceptable solutions much earlier using the enumeration algorithm of Raghavan *et al.* [Ragh81] as the base. Such limits on the crossover bound place a restriction on the maximum allowable number of crossovers but do not produce a minimum crossover solution. In fact the lack of any formal work on minimum crossover routing calls for the introduction of the following terminology quite similar to that used for channel via minimisation.

The problem of *crossover minimisation* in single row routing could be formulated as the *Constrained* crossover minimisation problem or the *Unconstrained* crossover minimisation problem. In a constrained crossover minimisation problem, an SRR layout is assumed to be available to start with and the goal is to reduce the number of crossovers by manipulating the nets and existing crossovers. On the other hand, UCM does not impose any restrictions on the topology and considers the SRR routing and crossover minimisation as an integrated step. Quite clearly, constraint crossover minimisation is analogous to the constrained via minimisation for channels and UCM is analogous to the UVM for channels. Again these definitions should consider the environment of the SRR problem as well. When the SRR is considered as an isolated problem, the constrained crossover minimisation and unconstrained crossover minimisation problems would be called CCMSRR and UCMSRR respectively. However, the different role of SRR in the SRT environment calls for the additional classifications of CCMSRT and UCMSRT. Thus, to summarise, the new problems concerning minimum crossover routing of SRR problems are as follows:

- CCMSRR Constrained Crossover Minimisation in isolated SRR case.
- UCMSRR Unconstrained Crossover Minimisation in isolated SRR case.
- CCMSRT Constrained Crossover Minimisation in SRT environment.
- UCMSRT Unconstrained Crossover Minimisation in SRT environment.

Although the above four problems are newly defined, it is clear that they are closely related to the channel via minimisation problems. Thus, it becomes imperative that one should draw from the extensive research done on channel via minimisation [Hsu83, Sadw84, Sarf89 *etc.*] This is possible due to the comparable properties between a channel and its corresponding SRR using the *abcd* type Forward Transformation; for example, the circle graph  $C_g$  of the channel is identical to

the overlap graph  $G_o$  of the SRR. Via minimisation in channels was also discussed in Chapter 2, under the constrained and unconstrained models. Since the emphasis in this chapter is on the unconstrained crossover minimisation using integrated reduction techniques, the analogous problem of Topological Via Minimisation of channels is briefly revisited.

## 6.2 Topological via minimisation of channels

TVM of channels was proposed by Hsu [Hsu83] for a 2-layer, 2-pin net case. Before this work, all work on via minimisation was on constrained via minimisation, where an initial channel layout was assumed to be available and was improved further (by layer reassignment of nets or net segments of nets without destroying the topology of the input realisation), to achieve fewer vias. Hsu proposed a new approach which handles the channel routing and the via minimisation problem at the same time from a global point of view. The approach consists of two major steps, namely, *topological routing* and *geometrical mapping*. The aim of topological routing is to find a minimum via topological solution. The geometric mapping process then maps the topological solution to the geometric plane conforming to the design rules. Basically for any channel routing problem there are two constraints, one of which is the *net intersection* constraint. Given a channel problem, the sequence of terminals along the boundary uniquely defines the intersections between the nets which in turn determines the lower bound on the number of vias. The other constraint is the *geometrical constraint* imposed by the design rules which set a lower bound on the area needed for any realisation. This approach attacks the two constraints separately but globally. Thus, given a channel problem, *topological routing* finds a topological solution with a minimum number of vias and is also referred to as *minimum via topological routing* or as unconstrained via minimisation.

Unconstrained via minimisation of two layer channels with only 2-pin nets has been shown to be NP-complete [Sadw84, Sarf89]. Hence, heuristic algorithms are justified and have been proposed [Hsu83, Sadw84, Chan87]. The algorithms proposed in [Hsu83] and [Sadw84] capture the channel net intersection in terms of the circle graph  $C_g$  of the channel and find a maximal bipartite subgraph of  $C_g$  to get a largest subset of the given netlist which is routable in 2 layers without vias. Such sets of nets are called *solid nets* and the process of identifying them is called *solid nets identification* (SNI). Thus, the problem of finding a maximum bipartite subgraph, also referred to as the minimum node deletion problem, finds the minimum number of nodes that need to be eliminated from  $C_g$  so that the resulting subgraph is bipartite. The problem can also be seen to be equivalent to the problem of finding a *maximum 2-independent set* (2-MIS) of a circle graph. After the solid nets are identified, the victims of the node deletion operation, referred to as *residual nets*, need to be routed using the *residual net routing* (RNR) step. Again, for a given channel problem there may be more than one 2-MIS. For each 2-MIS, more than one bipartition is possible if the corresponding circle graph is unconnected, because each connected component can be coloured independently from the others. Some of them could be better than others for the subsequent residual routing. Choosing the best bipartition is called *optimal bipartitioning*. Thus, the minimum via topological routing of channels is achieved by the steps of solid nets identification and residual net routing.

Hsu [Hsu83] restricts the consideration to only 2-pin nets and uses the greedy method of approximating a 2-independent set as a sequence of two 1-independent set problems. It is to be noted that such a greedy approach need not result in an optimal solution. A 1-independent set (also simply referred to as an *independent set*) represents the largest subset of the given netlist which can be routed in one layer. More formally, an independent set of a graph  $G = (V, E)$  is a subset of the vertex set  $V$ , such that no two of its elements are adjacent. That is,

there is no edge in  $E$  which connects any two elements in the independent set. An independent set is called *maximal* if no more vertices can be added to it without destroying the independence property. A *maximum* independent set (MIS or 1-MIS) is a highest cardinality maximal independent set. Sadwoska [Sadw84] applies the maximal planarisation algorithm of a circle graph to find a largest bipartite subset. [Sadw84] also proved that in an optimal solution of a TVM problem, every net need have at most one via. However, such a routing usually calls for long detours and meandering of nets.

Chang and Du [Chan87] follow a graph theoretic approach based on the properties of  $C_g$ . They use the close relationship between cycles of odd length. A graph can be 2-coloured only if it is bipartite and a graph can be bipartite if and only if it has no odd-length cycles. Thus, all odd cycles of  $C_g$  should be identified and broken. However, it takes exponential time to generate all cycles of odd length. They also make the observation that, on an average, the number of 3-cycles is much larger than other odd cycles and thus it is enough for all practical purposes to identify and break all the three cycles. Also breaking a 3-cycle will break any 5-cycle, 7-cycle *etc.* which superscribes that 3-cycle. Thus, they propose a greedy algorithm to detect the 3-cycles and then to select a minimum number of vertices to break the 3-cycles. If the graph is still non-bipartite, then all the fundamental cycles are generated and a vertex involved in as many fundamental cycles as possible is deleted.

Xiong and Kuh [Xion88, Xion89] have formulated the via minimisation problem as a  $\{0,1\}$  integer programming problem subject to some constraints. There is a constraint for each cross point (point where two nets intersect). Minimising the number of cross points would lead to minimising the number of constraints in the integer programming formulation. They have proposed a divide and conquer method in which they keep on subdividing a circle graph  $C_g$  into two simpler circle graphs until a crossing free channel is reached. In the final topological routing each



net pair which intersects in the original channel will intersect only once, which is obviously the minimum crossing. Their successive subdivision approach is applicable to both two and multipin nets and their approach can handle both CVM and UVM of channels.

### 6.2.1 A new heuristic for minimum node deletion

Minimum via topological routing of a channel is done in two steps, namely, *maximum bipartition* and *residual net routing*. Using the lemma of Sadwoska [Sadw84] that every net in an optimal solution of a topological via minimisation has at most one via, it is evident that each residual net can be routed with exactly one via. That is, the minimum number of vias  $V_o$  is equal to the number of residual nets. However, this approach calls for rearranging the already routed nets and may cause many detours. If a channel cannot have detours, then the number of vias may exceed the number of residual nets as illustrated in Figure 6.2.

The *maximum bipartition* problem has been solved in different ways [Hsu83, Sadw84 and Chan87] as previously described. They all require a complex series of steps to identify the minimum nodes to be deleted. Here, a much simpler heuristic is proposed to choose directly the next most suitable victim to be deleted. The heuristic is repeatedly applied until the graph becomes bipartite. This simple and efficient new heuristic, called the *maximum overlap degree* (MOD) heuristic, is based on the properties of the overlap graph  $G_o$  of the SRR problem, which is identical to the circle graph  $C_g$  of the original channel.

For the overlap graph  $G_o$ , the term *overlap degree* of a node in  $G_o$  refers to the number of edges which are connected to that node. This is equivalent to the number of nets with which the given net overlaps. In general, a net which has a high overlap degree is more likely to cause many crossovers and hence is a good

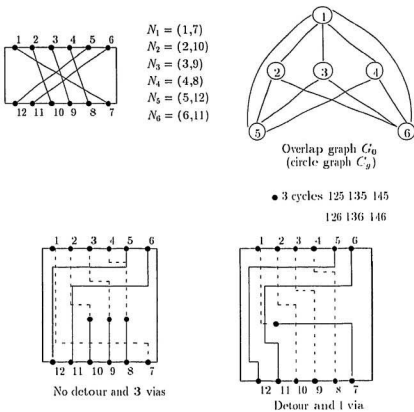


Figure 6.2: Topological via minimisation of a channel

candidate for deletion. This principle is the foundation of the proposed heuristic. Moreover, if a node  $N_e$  of  $G_o$  has many edges connected to it, then there is a very high probability that the node  $N_e$  will be a member of many 3-cycles and is therefore a good candidate for deletion.

The *overlap degree* of each node in  $G_o$  is found and the nodes are arranged in decreasing order of their overlap degree. The victims are then chosen from this list, starting with the node having the maximum overlap degree. After each deletion, it can be determined in linear time whether or not the subgraph is still bipartite. If not, the process is repeated. The proposed approach is illustrated in Figure 6.3 for a PNSRR problem and in Figure 6.4 for an MNSRR problem. Experience with different examples has led to the conclusion that the MOD heuristic often successfully identifies the victims (nodes to be deleted) and results in a minimum node deletion to get a maximum bipartite subgraph of the input problem. However, the fact that the proposed algorithm is heuristic clearly implies that a counterexample can always be contrived to show its failure. Such an example will be shown in section 6.3.5 and remedies suggested.

## 6.3 Routing of PNSRR problems

In this section, routing of a PNSRR problem is discussed. In a Permutation SRR problem all the nets are 2-pin nets and can be left, right or straight nets. Since the presence or absence of straight nets plays a role in routing, PNSRR problems are classified into two categories, namely, PNNSSRR problems which have no straight nets and PNYSRR problems which have at least one straight net. Routing of PNNSSRR problems is discussed first, highlighting different schemes. These schemes are then evolved for the PNYSRR problems.

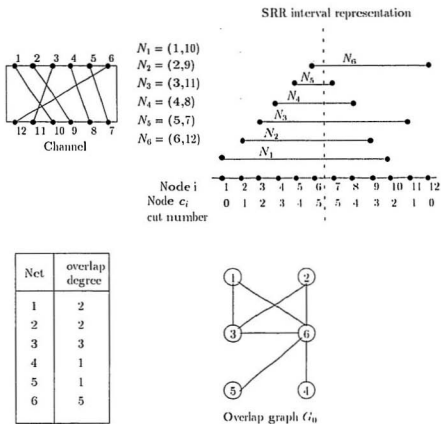


Figure 6.3: Node deletion of a PNSRR using MOD heuristic

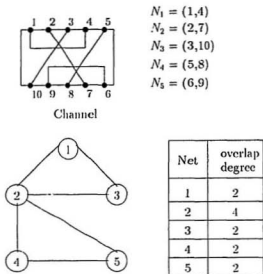


Figure 6.4: Node deletion of a MNSRR using MOD heuristic

### 6.3.1 PNNSRR with SNI+RNR approach

The approach of using the two distinct steps of *solid nets identification* and *residual net routing* as done in minimum via topological routing of channels, is applied to route the PNNSRR problem. The fact that topological via minimisation of permutation channels has been solved in  $O(n^2 \log n)$  [Sarf89, Rim89] time, suggests that the similar problem of minimum crossover routing of the Permutation SRR problem is likely to have a polynomial algorithm along the same lines. The solid nets identification process can be carried out by first drawing the *overlap graph*  $G_o$  of the PNNSRR problem and then finding a maximum 2-independent set of  $G_o$ . The MOD heuristic previously discussed will be useful in finding such a set. Once the bipartite subset has been identified, then the conceptual realisation for this subset can be obtained in the same manner as explained in Chapter 6. This will complete the solid nets routing step.

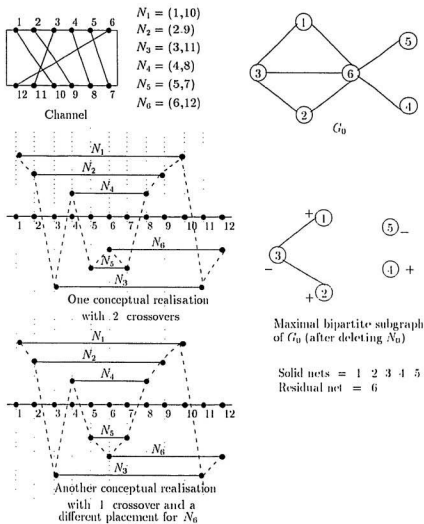


Figure 6.5: Routing of a PNNSRR using SXI+RNR approach

The next step is the residual net routing, which calls for a placement of the residual net(s) in the conceptual realisation such that upon conversion to final physical realisation by drawing the reference line and the topological mapping, the solid nets would still remain solid and the crossover(s) would be on the residual nets only. This restriction cuts down the number of possible placements for a given residual net. Let  $N(i)$  represent the net to which the node  $i$  belongs. Consider a residual net  $N_r$  whose left node is  $i$  and right node is  $j$ . Now the placement of  $N_r$  is influenced at its left end  $i$  by the placement of the nets  $N(i-1)$  and  $N(i+1)$  and the placement of  $N_r$  is influenced at its right end  $j$  by the placement of the nets  $N(j-1)$  and  $N(j+1)$ . In order that no crossovers are generated on any of the solid nets, the left end constraint is that the level of  $N_r$  should be between the levels of  $N(i-1)$  and  $N(i+1)$  and right end constraint is that the level of  $N_r$  should be between the levels of  $N(j-1)$  and  $N(j+1)$ . The different possible levels for placement of  $N_r$  are illustrated in Figure 6.6. Although in many cases the level of  $N_r$  can be found to satisfy both the left and right constraint, in some cases it may not be possible. Then a decision has to be made as to which constraint is to be satisfied and at what level the residual net would be placed within the range allowed by the satisfied constraint. This situation becomes more complex when there are multiple residual nets. This suggests that coping with the task of separate residual routing is non-trivial even for the PNSSRR problem, which is the simplest subproblem of the general non-bipartite SRR problem. This implies that the steps of SNI and RNR should be combined in some way so that a long and complex process is avoided for the residual nets placement and routing.

Residual nets result from the maximum bipartite subgraph process and the subsequent 2-colouring process. At the end of this process, all the solid nets have been assigned a (+ or -) colour, whereas the residual nets have no colour assigned yet. However, if LR-colouring instead of 2-colouring is used then all the nets would have had a (+ or -) label. Thus with LR colouring the street assignment can be

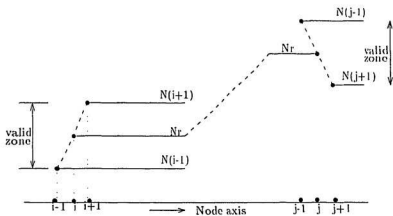
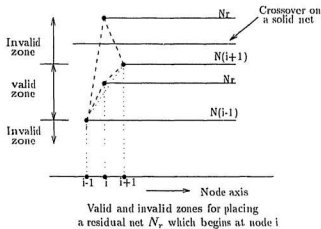


Figure 6.6: Placement of a residual net



fixed for all the nets of a PNNSRR. An additional order is still needed to place the nets at a particular level within the chosen street. This additional order could be either the box order or Tarn's order. The combination of LR-colouring and box ordering is an extension of the box procedure discussed in Chapter 5 and the combination of LR-colouring and Tarn's order is a modification of the TARN-MOD algorithm, with the 2-colouring process replaced by the LR-colouring. These schemes will be discussed now in more detail.

### 6.3.2 Extension of Box-procedure to PNNSRR

Solid nets identification by the 2-MIS approach and the subsequent 2-colouring of the bipartite graph leaves the residual nets uncoloured, whereas LR colouring uniquely colours all the nodes of a PNNSRR problem. The PNNSRR problem which was routed in Figure 6.3 using the SNI+RNR approach is routed with the box-procedure (LR-colouring and box-ordering) in Figure 6.7. Unlike the previous approach in which  $N_6$  was the residual net, here *all* the nets can be coloured depending upon whether they are left or right nets. The nets  $N_1, N_2, N_4$  and  $N_5$  are all right nets and the nets  $N_3$  and  $N_6$  are left nets. Using this LR-colouring for street assignment in the conceptual realisation<sup>1</sup> the nets  $N_1, N_2, N_4$  and  $N_5$  should be placed on one side (say upper) of the conceptual realisation and the nets  $N_3$  and  $N_6$  should be placed on the other (lower) side. However, to decide the particular level of placement within the chosen side, box ordering is used. It is to be recalled that such ordering sorts the nets in decreasing order of their starting point. This ordering will mean that the nets  $N_1, N_2, N_4$  and  $N_5$  are placed in the order of  $N_5, N_4, N_2$  and  $N_1$  in the pseudo tracks  $T_{-1}, T_{-2}, T_{-3}$  and  $T_{-4}$  respectively. Similarly,  $N_6$  will be placed in  $T_{+1}$  and  $N_3$  will be placed in  $T_{+2}$ . This conceptual realisation turns

---

<sup>1</sup>In the case of bipartite SRR problems, the conceptual realisation could be completely dispensed with whereas for the non-bipartite SRR problems the conceptual realisation and topological mapping (by reference line drawing procedure) is necessary to get the physical realisation.

into a physical realisation with one crossover, which is the minimum number of crossovers for this example. In fact, the overlap of left-right type (LR crossing in the channel) does not contribute to crossovers because  $L$  and  $R$  nets are assigned to opposite layers, whereas the overlap of left-left or right-right type (LL or RR crossing in the channel) contribute to crossovers. Thus, the number of crossovers is at least equal to the sum of LL and RR crossings. In the example, there is only one LL crossing (between  $N_3$  and  $N_6$ ) and hence at least one crossover is needed for any realisation of this SRR problem.

Using the box procedure, every LL or RR crossing would result in one crossover. When there is an LLL or RRR crossover, this results in a double crossover, as illustrated in Figure 6.8. This problem requires at least two crossovers and thus the box procedure produces a minimum crossover solution for the multiple crossover problem as well. The double crossover can be translated into two vias using the multiple crossover handling techniques to be discussed in Chapter 7.

### 6.3.3 Extension of TARNG-MOD to PNNSRR

In this scheme LR-colouring combined with Tarnig's ordering is used to route a PNNSRR problem. That is, Tarnig's order instead of the box order is used to decide the placement of the nets within the chosen street in the conceptual realisation. Considering the same example as shown in Figure 6.7, the zone allotment process is as shown in Figure 6.9. This leads to the Tarnig's ordering (5,6),4,2,(1,3). Now nets  $N_1, N_2, N_4, N_5$  are all left nets and based on Tarnig's order, the left nets will be assigned in the order  $N_5, N_4, N_2$  and  $N_1$  which will place them in the pseudo tracks  $T_{-1}, T_{-2}, T_{-3}$  and  $T_{-4}$  respectively, in the conceptual realisation. Similarly, for the right nets  $N_3$  and  $N_6$  the above Tarnig's order stipulates that  $N_6$  should be routed before  $N_3$ . This results in the placement of  $N_6$  in track  $T_{+1}$  and  $N_3$  in track  $T_{+2}$ . The overall conceptual realisation is then no different from that obtained by

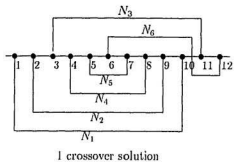
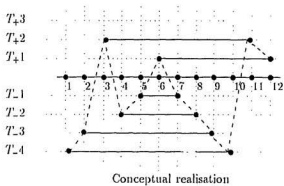
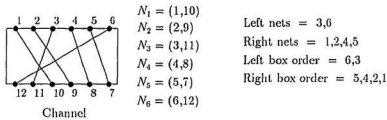


Figure 6.7: Routing a PNNSRR problem by box procedure

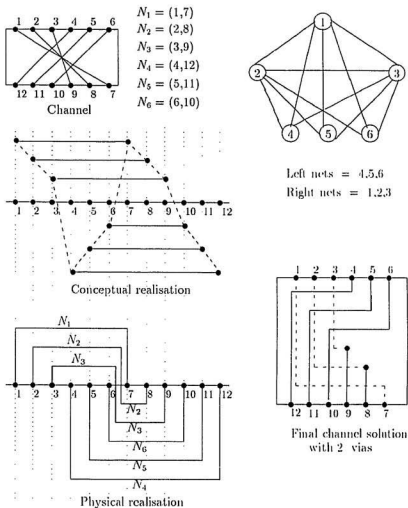
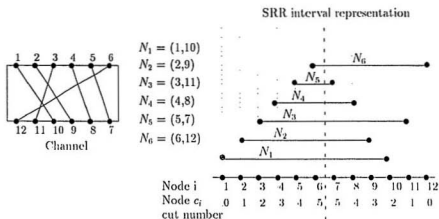


Figure 6.8: PNNSRR problem with a RRR crossing routed by box procedure

the alternate procedure of LR-colouring together with box ordering. However, this is not true, in general.

The following presents some new results regarding Tarng's ordering for a PSRR problem. In a PSRR, since all nets start before any net ends, PSRR is a single group SRR. Thus, any two intervals either overlap or contain and thus the overall interval graph  $G_I$  is a clique. Further, as the nodes are scanned left to right, the cut numbers first uniformly increase until the middle node and then uniformly decrease. This is seen in Figure 6.9. This increasing and decreasing sequence of the cut numbers of nodes results in an analytical expression for the cut number of any node. Consider a net  $N_i$ . Its cut number  $q_i$  is defined as the maximum of the cut numbers of its left and right nodes,  $C_{li}$  and  $C_{ri}$  respectively. Now if the nets are numbered based on their left nodes, then  $C_{li} = i - 1$  and  $C_{ri} = (2 * m - i - L_i)$ , where  $m$  is the total number of nets and  $L_i$  is the length of net  $i$ . Thus,  $q_i = \text{Max}\{C_{li}, C_{ri}\} = \text{Max}\{(i - 1), (2 * m - i - L_i)\}$ . Another interesting observation is that in a PSRR each zone can have at most two nets. This is because all the nets in the same zone must have the same cut number  $q$ , and for a PSRR, no more than two nets can have the same  $q$  value and hence no more than two nets can be present in the same zone. Thus, no residual ordering procedure is needed; just zone allotment would be sufficient to fix Tarng's order for the PSRR problems. Further, in the case of PNNSRR problems, for each zone which contains two nets, one should be  $L$  and the other should be  $R$ . For example, in the order given above, in the zone which contains  $(5,6)$ ,  $N_5$  is a left net and  $N_6$  is a right net.

In the example discussed in Figure 6.9, Tarng's order and box ordering were identical, which however is not always true. Figure 6.10 shows an example where the two orders are different. The order for the left nets, (i.e.,  $N_2, N_4$  and  $N_6$ ) is 6,4,2 using both Tarng's and box ordering. However, for the right nets (i.e.,  $N_1, N_3$  and  $N_5$ ), Tarng's ordering is 5,1,3 whereas the box ordering is 5,3,1. The example when routed with Tarng's order results in two crossovers whereas



Net $i$	Length $L_i$	Cut number $q_i = 2m - i - L_i$
1	9	2
2	7	3
3	8	2
4	4	4
5	2	5
6	6	5

Zone	Nets	$q$
$Z_0$	$N_5, N_6$	5
$Z_1$	$N_1$	4
$Z_2$	$N_2$	3
$Z_3$	$N_1, N_3$	2

Overall Tarnig's order = (5,6), (4,2), (1,3)  
 order                    - +       - -       - +

Right nets = 1,2,4,5       - Tarnig's order for right nets = 5,4,2,1  
 Left nets = 3,6            + Tarnig's order for left nets = 6,3

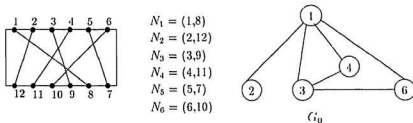
Figure 6.9: Tarnig's order based level assignment

the minimum possible crossover is one. This is where the modified definition of *minimum crossover* becomes useful. Since the two crossovers produced by Tarng's order are adjacent they could be collapsed into a single via on the channel.

While concluding this section on routing PNNSRR problems, it is clear that the LR colouring approach is superior to the 2-colouring. With the LR-colouring approach, the two schemes of LR-colouring combined with Tarng's ordering and LR-colouring combined with box ordering produce solutions with different numbers of crossovers, considering the absolute crossover count. However, such solutions may well be equivalent in the number of contacts in the channel. In general, it is found that LR-colouring combined with box ordering produces solutions whose crossover count is equal to or slightly greater than the minimum crossover count  $C_0$  for the problem. On the other hand, LR-colouring together with Tarng's ordering results in solutions with more crossovers, which, however are often collapsible. The difference between the two approaches is due to the fact that box ordering captures the containment relation more closely than does Tarng's ordering. Box ordering is purely dependent on the reverse ordering of left end points whereas Tarng's ordering is based on the cut number of nets which in turn depends on the cut number of both the left and right nodes. Apart from its higher probability for obtaining fewer crossovers, box ordering is a very simple process, leading to implementation advantages. Between the two schemes, LR-colouring combined with box ordering turns out to be the better choice.

### 6.3.4 PNSRR with straight nets (PNYSRR)

Routing a PNYSRR problem is the same as routing a PNNSRR problem, except that that the straight net(s) in the PNYSRR problem needs attention. Thus, the approaches discussed above for PNNSRR problems could be applied to PNYSRR problems, with suitable modifications. The straight nets need to be coloured  $L$  or



Left nets = 2,4,6

Right nets = 1,3,5

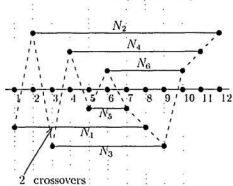
Box order = 6,4,2 for left nets

Box order = 5,3,1 for right nets

Targ's order = 5,1,3 for left nets

Targ's order = 6,4,2 for right nets

Targ's order = (5,6), (4,1), (3,2)  
 - + + - - +



Conceptual realisation using Targ's ordering

Figure 6.10: Routing a PNNSRR problem by TARG-MOD

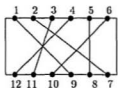


$R$  to fix the street assignment and then their placement within the street needs to be fixed.

Different options were tried for colouring the straight nets, namely, always colouring as left or always colouring as right or randomly. The placement level can be based on Tarng's order or box order or an order which places the straight net in such a level that crossovers are maximised on it. This exploits the fact that all the crossovers on a straight net are pseudo crossovers and can be collapsed into a single via, using suitable crossover handling techniques. Although the solution with many crossovers on the straight nets may look bizarre and unwieldy, it is essentially the same as another solution with one crossover on the straight net.

Figure 6.11 illustrates a PNYSRRT example which is routed with these different options. This example needs at least two crossovers, because the LL-crossing between nets  $N_3$  and  $N_4$  necessitates one crossover and the LRS (left-right-straight) crossing among the nets  $N_6$ ,  $N_5$  and  $N_2$  calls for one crossover. An LRS crossing can also be explained from the overlap graph  $G_o$ . Each LRS crossing results in a 3-cycle in  $G_o$  which needs at least one crossover. By looking at the different solutions it is clear that although some have two crossovers and others have three crossovers, they all can be considered equivalent in the light of the new definition of minimum crossovers. However, as in the case of PNNSRR problems, box ordering, in general, results in fewer vias than Tarng's ordering for the PNYSRRT problem as well. Thus, LR-colouring combined with box ordering turns out to be a good candidate for routing both PNNSRR and PNYSRRT problems.

Figure 6.12 illustrates an extreme case of crossover collapsing where multiple crossovers are loaded on the straight net. This solution is no worse than the alternative solution shown, which has fewer crossovers on the straight net. However such a wriggling straight net would result in extra storage if the SRR layout is stored physically (geometric coordinate representation) and not logically. Such



L left = 3,4,6

R right = 1,2

S straight = 5

Tarnng's order = (2,6), 5, (1,4), 3

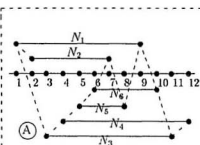
+ Right

- Left

$N_1 = (1, 9)$   $N_4 = (4, 12)$

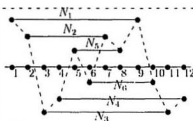
$N_2 = (2, 7)$   $N_5 = (5, 8)$

$N_3 = (3, 11)$   $N_6 = (6, 10)$



(A)

- $N_5 \rightarrow Left$
- Box order used 6,5,4,3 (Left) -  
2,1 (Right) +
- 3 crossovers



(B)

- $N_5 \rightarrow Right$
- Box order used 6,4,3 (Left) -  
2 crossovers 5,2,1 (Right) +

(C)

- $N_5 \rightarrow Left$
- Tarnng's order used  
(2,6), 5, (1,4), 3  
Left = 6,5,4,3 (-)  
Right = 2,1 (+)
- same as solution (A)

(D)

- $N_5 \rightarrow Right$
- Tarnng's order used (2,6), 5, (1,4), 3
- 3 crossovers Left = 6,4,3 (-)  
Right = 2,5,1 (+)

Figure 6.11: Routing a PNYSRR problem

wriggling may also pose problems in the Backward Transformation step. This can be avoided by not placing the straight net near the node axis, especially if its SRR length is large.

### 6.3.5 Maximum conflicting overlap degree heuristic for the minimum node deletion problem

The *LR* colouring of a PNSRR has applications in modifying the maximum overlap degree heuristic discussed earlier, to find the best possible victim to be deleted during the construction of the *maximum bipartite subgraph* of the given  $C_g$  or  $G_a$ . Figure 6.13 illustrates a PNSRR example where the maximum overlap degree heuristic fails. Such examples do not occur frequently in practice. Here the maximum overlap degree heuristic identifies the net  $N_3$  as the victim because it has the highest overlap degree of 4. Breaking  $N_3$  is not useful because it is not a member of any odd cycle despite its many radial links. The reason for non-bipartition is the odd cycle formed by nets  $N_1, N_2$  and  $N_4$  and one of them must be deleted to break the cycle.

The failure of the MOD heuristic (choosing  $N_3$  as the victim) suggests that a victim should not be deleted if it is not a member of any odd cycle, but an explicit verification of such a process is complex. Instead, two other approaches suggest themselves. In one approach, the deletion follows the MOD heuristic and would delete  $N_3$  and then  $N_4$  as residual nets and declare  $N_1, N_2, N_5, N_6$  and  $N_7$  as solid nets. At the end of the process, an attempt is made to reintroduce each of the residual nets as a solid net, if such an introduction still leaves the graph bipartite. Thus,  $N_3$  would be converted from a residual net to a solid net. However, this approach has some extra and often unnecessary work. The other approach is more elegant and is based on the classification of the overlap edges as *conflicting* or *non-*

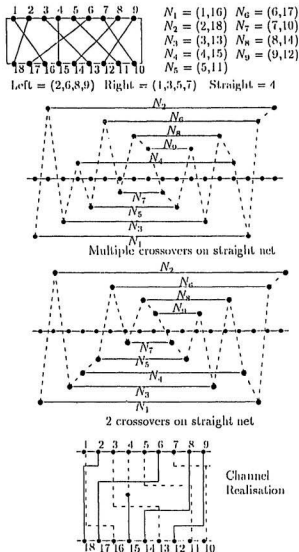


Figure 6.12: A PNYSRR problem with multiple crossovers on the straight net

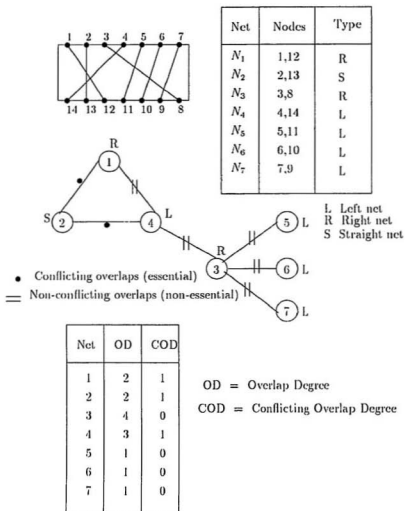


Figure 6.13: An example where MOD heuristic fails

*conflicting*. The resulting heuristic is called *maximum conflicting overlap degree heuristic*, where the victims are chosen using conflicting overlap degree instead of the total overlap degree. This approach will be explained now.

It was already mentioned that LR type crossings are of no consequence in routing a PNSRR because  $L$  and  $R$  nets are routed on opposite layers. The overlaps which necessarily lead to a crossover are called conflicting overlaps. LL and RR belong to this type whereas LR is a non-conflicting overlap. Again, LLL can be visualised as three LL relations out of which only any two would necessitate crossovers. The same concept is extendible to multiple L crossings. The LRS crossing can be split as LS, RS and LR crossings. The LR crossing is not essential whereas the LS and RS are. For the node which represents the straight net the conflicting overlap degree is still one because breaking either the SL link or the SR link breaks the odd cycle corresponding to the LRS crossing. Using this approach each of the overlap edges of an overlap graph  $G_o$  can be classified as conflicting (essential) or non-conflicting (non-essential). Such a classification is shown in Figure 6.13 where the conflicting overlap edges are marked as '•' and the non-conflicting overlap edges are marked as '='. Using this approach and counting only the conflicting overlap degree,  $N_3$  will no longer be chosen. The LRS crossing of the nets  $N_1$  (R),  $N_2$  (S) and  $N_4$  (L) results in the odd cycle 1-2-4 and the conflicting overlap degree of all the three nodes is one and thus any one of them can be broken. This will make the graph bipartite with 1-node deletion, which is the minimum for this example. Many cases for which the maximum overlap degree heuristic fails can often be handled by the maximum conflicting overlap degree heuristic. When multiple nodes need to be deleted as the deletion process progresses, a link which was conflicting type may not be so any longer. For example, in an LRS cycle, both the LS and RS links are to be considered essential initially. After deleting one of them the cycle is broken so that the other becomes non-essential. Therefore, some kind of dynamic status must be maintained for the links.

### 6.3.6 Logical routing of PNSRR

The preceding classification of overlap edges into *conflicting* and *non-conflicting* leads to an interesting way to route logically a PNSRR. The approach is called *logical* because unlike the methods in which the layout information is represented in explicit geometric form, a graph is constructed which essentially contains the same information as an explicit geometric layout. The idea is also motivated by the fact that if the SRR is bipartite, then the SRR layout step can be bypassed altogether. That is, an explicit SRR layout need not be constructed and then processed through a Backward Transform step. The channel realisation can be obtained directly based on simple calculations to establish the relation between the SRR tracks and channel tracks. Such a non-geometric Backward Transformation is highly suitable for a programming implementation as well.

If the non-bipartite graph is somehow transformed into a bipartite graph, then the routing can proceed in the same way as for a bipartite problem. Therefore, the goal is to transform the non-bipartite  $G_o$  of a PNSRR into an *equivalent* bipartite SRR. This will call for introducing *pseudo nodes* in the original  $G_o$ . For example, consider a  $G_o$  which is non-bipartite due to the presence of a 3-cycle. The original approach (SNI + RNR) of pulling out a node as a residual node by the node elimination process and then re-introducing it through the residual net routing process, clearly introduces extra work. A better alternative is to keep all the original nodes intact, but to introduce pseudo nodes along the links that need to be broken. By introducing a pseudo node on any one of the three links that make up an odd cycle, an odd cycle will turn into an even cycle and the graph will become bipartite. This process of introducing pseudo nodes on selected edges of the overlap graph is called *logical splitting*. Each such node on the broken link is *pseudo* in the sense that it corresponds to introducing a dummy node on one of the two nets connected by that link. For example, a pseudo node introduced on

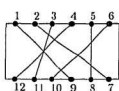
the edge between 3-4 in the overlap graph, can turn into a dummy node on net  $N_3$  or  $N_4$ , corresponding to a crossover on net  $N_3$  or net  $N_4$ . This breaks a 2-pin net into a 3-pin net. Such a process is referred to as *net splitting*. The third and middle node is essentially the point at which the crossover takes place. Thus, the location of the pseudo node between the real end nodes of the net to be split is to be determined based on the node interval at which the crossover would take place. This process of deciding the exact location of the pseudo node on a logically split net is called *physical splitting*.

Thus, logical splitting will specify which net will be split and thereafter the physical splitting will decide where exactly it is split. These two pieces of information will be enough to specify which nets have crossovers as well as where the crossovers are located. Each logical splitting process breaks a conflicting overlap, and this process must be repeated until no more conflicting links are present. After all the logical splitting is complete, the physical splitting process has to proceed. The overall approach is illustrated for a PNYSRR problem in Figure 6.14.

Given a PNSRR problem, first its overlap graph  $G_o$  is drawn. Then each of the overlap edges is classified as conflicting or non-conflicting. For the example shown in Figure 6.14, there is a conflicting link between nodes 3-4 of type LL and an LRS cycle between nodes 2,5 and 6 which would translate into an LS and an RS link, one of which needs to be broken. Let the candidates for logical splitting be links 3-4 and 2-5. Again the splitting of 3-4 can be translated as a crossover on  $N_3$  or  $N_4$ . Let  $N_4$  be chosen. Thus a new node labelled  $4^1$  will be introduced on the link 3-4 making it 3- $4^1$ -4. This would turn two odd cycles (1-3-4 and 2-3-4) into even cycles.

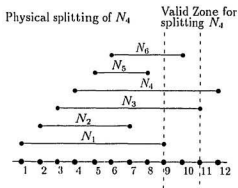
Now the physical splitting of  $N_4$  and  $N_5$  has to take place. In general the physical splitting has to be decided such that no new odd cycles will be formed, because such a formation is clearly counter to the goal of constructing a bipartite



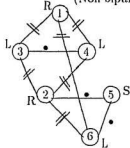


$N_1 = (1,9)$   
 $N_2 = (2,7)$   
 $N_3 = (3,11)$   
 $N_4 = (4,12)$   
 $N_5 = (5,8)$   
 $N_6 = (6,10)$

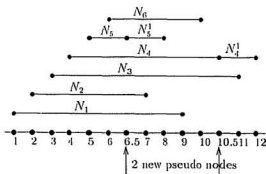
L Left nets = 3,4,6  
 R Right nets = 1,2  
 S Straight net = 5



Original overlap graph  $G_0$   
(Non-bipartite)



Modified interval representation



Modified overlap graph  
(Bipartite)

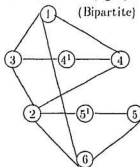


Figure 6.14: Logical routing of a PNYSRR problem

graph. Thus, when the original net  $N_4$  is split into two nets 4 and  $4^1$ , some of the overlap relations of  $N_4$  will turn into containment relations but no containment relation of  $N_4$  should turn into a overlap relation. These constraints on the pseudo node help to converge onto one or more possible locations for the pseudo node placement. For example, the fact that node  $4^1$  should not overlap with the net  $N_1$  in the overlap graph and the fact that net  $4^1$  has to start between the nodes 4 and 12 dictates that  $4^1$  has to start after node 9 which is the right end node of  $N_1$ . The choice can be further narrowed down based on other constraints and the location 10.5 can be finally chosen. However, such a unique location is not possible when there are multiple possible locations for a crossover placement. This is because for a given SRR problem there may be many different crossover solutions. A given SRR problem with many conflicting links can be broken in different ways leading to different logical splitting, each of which eventually translates into solutions with the same number of crossovers. Hence it is not necessary to differentiate between them unless there is a specific requirement such as avoiding the crossovers on some specific nets (say critical nets whose operating characteristics demand no via can be placed on them). Further, physical splitting of one net will influence the physical splitting of the other nets. This is because without fixing the physical splitting, the overlap relation between the the dummy nodes and others cannot be determined. When there are only a few crossovers this process can easily converge, but potential difficulties arise when the interaction between multiple constraints must be considered. Although this approach looks promising, more research is needed on this issue before it becomes a feasible, practical approach.

To conclude this section on PNSRR routing, it is observed that while many approaches were proposed to route Permutation non-bipartite SRR problems, LR-colouring combined with the box-ordering procedure turns out to be a good candidate and hence would be chosen to route any PNSRR problem and would form the core of the SRR solver step for non-bipartite cases.

## 6.4 Routing of MNSRR problems

The discussion so far has indicated that the order of increasing complexity in Single Row Routing is PBSRR, MBSRR, PNNSRR and PNYSTRR respectively. The MNSRR which is the last member of the series of increasing complexity is the most general SRR problem, and thus the most difficult to route with minimum crossovers. In fact, the general problem of crossover minimisation is NP-complete [Tsuk80]. Simply extending the different methods discussed for PNSRR to MNSRR proves to be a failure. The approach which is recommended based on its practicality as well as its effectiveness exploits a divide and conquer strategy. Here an MNSRR problem is split into a number of PNSRR problems. Then each PNSRR problem is routed, with the solutions of the subproblems then merged to achieve the solution of the overall MNSRR problem. However, such merging satisfying all the subsolutions may not be possible in all cases.

### 6.4.1 MNSRR by extending previous approaches

It is obvious that MSRR routing is more difficult than PSRR routing. This is because, as seen in the section on MBSRR problems, the important properties which are possessed by a permutation channel are lost in a mixed channel. In the absence of such convenient properties designing the routing algorithms becomes a difficult and challenging task. The main reason for the difficulty in MSRR routing stems from the presence of the *local* nets.

The approach of solid net identification and subsequent residual net routing (SNI+RNR) which proved to be difficult for the PNSRR problem is even more difficult for the MNSRR problems. The problem is, as before, in the residual net routing phase. The LR-colouring approach which was used for PNSRR case fails

for the MNSRR case because the philosophy of keeping  $L$  and  $R$  nets in opposite streets is no longer valid for an MSRR problem, as explained previously. Further, the coupling between local nets through the permutation nets forces some restrictions on the street assignment. For an MNSRR problem neither the 2-colouring nor the LR-colouring approach is adequate to solve the street assignment problem. In the absence of such an assignment neither Tarng's ordering nor box ordering can work. Thus, this approach is rejected and a more practical approach based on a divide and conquer philosophy is discussed next.

#### 6.4.2 Routing of MNSRR by grouping into PNSRR

This approach was motivated by a proposal by Du and Liu [DuLi87] to extend the applicability of Tarng's [Tarn84] algorithm, in the context of minimum track routing of SRR problems. Du and Liu proved that Tarng's algorithm results in an optimally congested SRR solution only for the single group SRR problems and may or may not produce optimal solution for multi-group SRR problems. They proposed that a multi-group SRR problem be split into a number of single group problems and each of these subproblems be routed using Tarng's procedure. Then, an attempt is made to merge all the subsolutions, without violating the order demanded by each of them. Such merging is not always possible because the adjacent groups have common nets and can demand conflicting orders. However, when the interaction between adjacent groups is low then the chances of overall merging is better. If the overall merging is a failure, Du *et al.* [DuLi87] proposed to use a left to right process, in which the merging operation does not honour all the sub solution constraints.

It is now proposed that such a grouping process be applied to manage the complexity of minimum crossover routing in MNSRR. The success of the grouping approach in optimally congested routing of a general SRR problem and the fact

that LR-colouring combined with box ordering approach is available to route a PNSRR problem, makes this divide and conquer approach practical and effective. Further, our experience has shown that it is difficult to come up with a single and comprehensive heuristic which is quite effective for the minimum crossover routing of MNSRR problems which also clearly favours the problem reduction approach.

Thus, an MNSRR problem which, in general, is a multi-group SRR problem, is first split into a number of PNSRR problems. Such splitting into groups can be done by identifying the maximal cliques [Sher89] of the overall interval graph  $G_I$  of the MNSRR problem. A PNSRR is a single group SRR and its  $G_I$  is a clique. Figure 6.15 illustrates an MNSRR example which is split into two PNSRR problems, (PNSRR1 and PNSRR2) and routed. This subdivision process also converts an originally local net into a permutation net. For example, the local net  $N_2$  of the MNSRR problem turns into a right net in the PNSRR1 problem. This conversion is the key point which greatly reduces the problem of routing local nets faced in the original MNSRR problem. For each PNSRR, a conceptual realisation is obtained and then adjacent realisations are merged to get the overall realisation. As mentioned earlier, such merging may fail if the clique intersection [Sher89] is quite high, and in that case the approach of merging adjacent groups into pseudo cliques can be pursued.

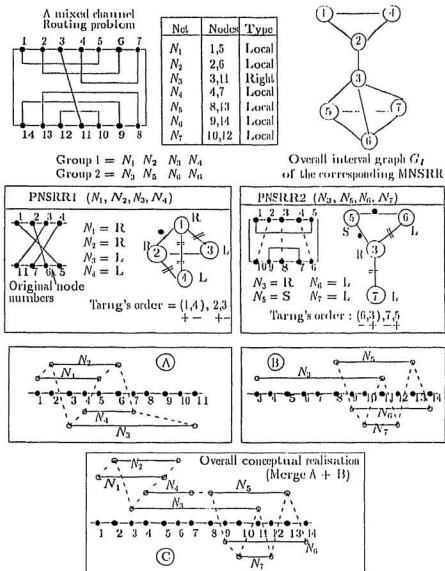


Figure 6.15: MNSRR routing by grouping

## Chapter 7

# Backward Transformation



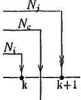
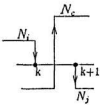
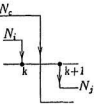
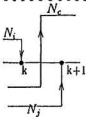
This chapter discusses in detail the Backward Transformation (BT) of the *abcd* type SRT. The purpose of the BT is to fold the SRR layout back into a channel layout, where the SRR layout is crossover free in the case of bipartite SRR problems and has crossovers in the case of non-bipartite SRR problems. The BT consists of two steps, namely, *crossover handling* and *fold back*. The crossover handling step manages those crossovers which remain after the reduction process. This step will be the front end of BT and deals with the problem of *crossover alignment*. Two techniques called *simple-align* and *split-align* are discussed. Then a generalised crossover handling technique called *grid opening*, which is capable of handling any type of crossover, is proposed and discussed. Finally, the fold back step is discussed with reference to channel width reduction under the 2-way overlap model with knock knees.

## 7.1 Crossover Handling techniques

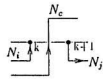

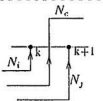

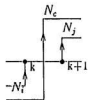
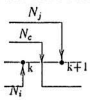
This step acts as the front end of the BT process and is concerned with the task of aligning those crossovers that are unreduced and are present in the SRR solution. Crossover handling is necessitated by a basic difference between the SRR and channel models, namely, the SRR model allows crossovers in the inter-node intervals while the channel model allows vias only at grid points. The term *aligning* of a crossover refers to the process by which a crossover is shifted from its inter-node position to a grid aligned position, so that upon folding back, the corresponding via would be located at a grid point. In this section, two techniques for crossover alignment are discussed. The first is the simple-align technique which is suitable to handle easy crossovers and the second is the split-align technique which is needed for handling the difficult crossovers. The classification of a crossover as easy or difficult depends on its ease of alignability, as explained below.

Consider a net  $N_e$  which has a crossover between nodes  $k$  and  $k + 1$  of an SRR realisation. Let the node  $k$  belong to net  $N_i$  and the node  $k + 1$  belong to net  $N_j$ . To align the crossover of the net  $N_e$ , the best choices are the nearest grids on the left or the right, that is, the grids at node  $k$  or  $k + 1$ . Whether such an alignment is possible or not depends upon the nature of nets  $N_i$  and  $N_j$ . Nets  $N_i$  and  $N_j$  can run in upper or lower streets and begin or end at nodes  $k$  and  $k + 1$  leading to 16 different combinations. The eight combinations in which the net  $N_i$  ends at node  $k$  (called class-A) as illustrated in Figure 7.1, pose no problems for alignment and are easy crossovers. To align an easy crossover, the crossing over net is split at node  $k$  or at node  $k + 1$ . Further, in order to make the end points of the split levels of the net coincide at a via after folding back, lateral shifting of other nets may be needed and this may lead to an increase in the number of tracks. An example of aligning an easy crossover is illustrated in Figure 7.2. The remaining eight combinations called class-B, where the net  $N_i$  starts at node  $k$ ,



Nets		Crossover Patterns		Type:
$N_i$	$N_j$			
UE	UB			Easy
UE	UE			Easy
UE	LB			Easy
UE	LE			Easy

UB Upper street Beginning net    LB Lower street Beginning net  
 UE Upper street Ending net    LE Lower street Ending net

Nets		Crossover Patterns		Type:
$N_i$	$N_j$			
LE	LB			Easy
LE	LE			Easy
LE	UB			Easy
LE	UE			Easy

UB    Upper street    Beginning net  
 UE    Upper street    Ending net  
 LB    Lower street    Beginning net  
 LE    Lower street    Ending net

Figure 7.1: Relation between crossover and adjacent nets - Class A

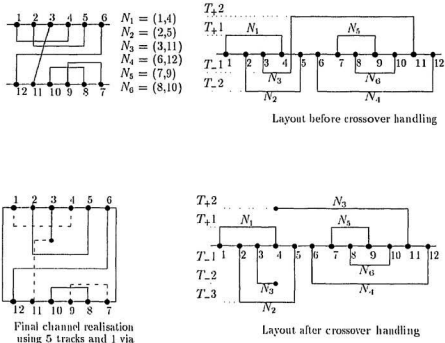










Figure 7.2: An example of aligning an easy crossover

as shown in Figure 7.3, can be easy or difficult or impossible. Thus, out of the sixteen combinations, two are impossible, twelve are easy and two are difficult. The analysis can be extended to higher crossover bounds as well. Consider, for example, a case with a crossover bound  $K = 2$  (two nets crossover in the interval between nodes  $k$  and  $k + 1$ ). The alignment process is illustrated in Figure 7.4. It is clear that the easy crossovers can be grid aligned to the nearest left or right grid. This process is local in the sense that it changes the layout of only the crossing over net and only pushes other nets laterally, if needed, but does not disturb them otherwise.

While the simple-align technique is successful in handling easy crossovers, it fails for the difficult crossovers because there is no way of aligning the crossing over net at node  $k$  or  $k + 1$ . This happens because the adjacent nets  $N_i$  and  $N_j$  cover the crossing over net  $N_c$  in both top and bottom; this is called a *full covering*. On the other hand, the easy crossovers are *half covered* or *full open*, because the nets  $N_i$  and  $N_j$  cover  $N_c$  only partially or not at all. Thus, aligning difficult crossover calls for the split-align technique where the split segments of the crossing over net  $N_c$  are restored to the same street to eliminate the crossover. But this flipping of  $N_c$  would interfere with one or more nets and each such crossing needs to be handled. The decision as to whether  $N_c$  should be restored to upper street or lower street depends on two factors, namely, the number of nets interfered and the depth of affection. Consider the difficult crossover shown in Figure 7.5. If the net  $N_2$  were flipped to upper street totally then it would interfere with only  $N_3$  whereas if it were flipped to lower street it would interfere with three nets  $\{N_1, N_5$  and  $N_4\}$ . Thus,  $N_2$  should be flipped to the upper street. Further, if the interfered net runs deep (*i.e.*, in a track far from the node axis), then to make its split levels coincide at a via, more tracks need to be pushed down. For example, the net  $N_3$  which was running at track  $T_{+2}$  demands that its newly added vertical piece project up to track  $T_{-2}$  so that its segments will match properly after folding back. This necessitates pushing

$N_i$	$N_j$	Crossover patterns	Type:	$N_i$	$N_j$	Crossover patterns	Type:
UB	UB		Easy	LB	UB		Easy
UB	UE		NP	LB	LE		Diff
UB	DB		Easy	LB	LB		Easy
UB	LE		Diff	LB	LE		NP

UB Upper street Beginning net  
 UE Upper street Ending net  
 LB Lower street Beginning net  
 LE Lower street Ending net

NP - Not possible  
 Diff - Difficult

Figure 7.3: Relation between crossover and adjacent nets - Class B

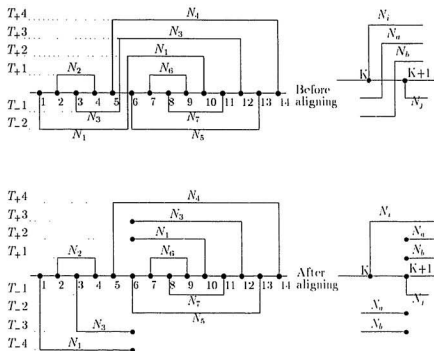


Figure 7.4: An example of aligning an easy 2-bound crossover

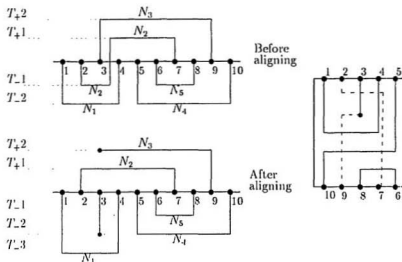


Figure 7.5: Handling of a difficult crossover by split-align technique

the net  $N_1$ , which was previously at track  $T_{-2}$ , to track  $T_{-3}$  and increasing the street width. If the interfered runs deeper originally, more track pushing will be called for. The split-align technique could result in a large increase in the number of vias as well as in the number of tracks needed. Further, the split-align technique may call for detours<sup>1</sup> and *external doglegs*<sup>2</sup> to join up the segments of the detouring net. Such an example is illustrated in Figure 7.6 where net  $N_2$  is forced to have a detour and external doglegs. Incorporating such detours and doglegs make the BT process more complex. Consequently, a more general crossover handling technique is needed which would take care of any type of crossover effectively and for this purpose a new alignment technique called the *grid-optic* technique will be discussed in section 7.2.

<sup>1</sup>A detour of a net in a channel is any horizontal run of the net outside its span.

<sup>2</sup>A dogleg is external if it occurs in a column outside the span of the net.

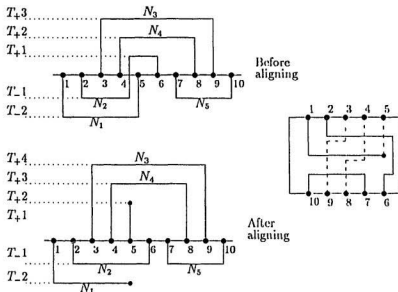


Figure 7.6: An example of split-align that needs an external dogleg

### 7.1.1 Crossover collapsing

The crossover handler should be intelligent in the sense that it should be able to detect and effect crossover collapsing described in Chapter 6. Such collapsing would translate multiple pseudo crossovers as well as adjacent real crossovers into a single via in the channel. To effect such collapsing, first the pseudo and real stretch of each net with crossovers should be identified. Basically any net can be considered to make up a *left real* piece, a *middle piece* and a *right real* piece. For a straight net the entire middle piece is pseudo whereas for the left and right nets a part of the middle piece is pseudo. For local nets the entire middle piece is real. When the channel is formed after the folding back only the real stretches should show up.

The crossovers on a straight net need special attention, because straight net crossover collapsing is a special case of pseudo crossover collapsing. Any number



of crossovers on a straight net could be translated into a single via as the entire stretch of a straight net is pseudo (whereas, for the left and right nets both real and pseudo stretches are present necessitating separate collapsing for each). If no detour is allowed in the channel, then the straight net can be replaced by just two vertical pieces running in opposite layers which connect through a via after BT. Trying to handle crossovers on a straight net is unnecessary as well as troublesome as illustrated in Figure 7.7. Here, a given SRR layout with a single crossover on the straight net  $N_4$ , is handled in two different ways. In one case, the straight net crossover is handled by the split align technique leading to solution with two vias which also calls for a detour on net  $N_4$ . In the other case, the straight net  $N_4$  is simply replaced by two vertical pieces of wires and this leads to a solution with one via. Thus, it is clear that the quality of the final solution depends on the handling technique used.

## 7.2 Grid opening: A generalised crossover handling technique

The split-align crossover handling technique could fail for some difficult crossovers and could necessitate long detours of nets. This is seen from the examples shown in Figure 7.6 and Figure 7.7. With such detours, a net will lie in more than one horizontal track, and doglegs are needed to connect the horizontal segments. Incorporating detours in the BT makes it more complex and also has the problem of choosing the optimal location for doglegs. An alternate technique which would effectively free up the congested grid as done by detouring is needed. This technique should also be general and applicable to any crossover. Such a general handling technique ensures that an SRR layout with any type of crossover can be folded back into a legal channel layout, thereby giving 100% success in routing any in-

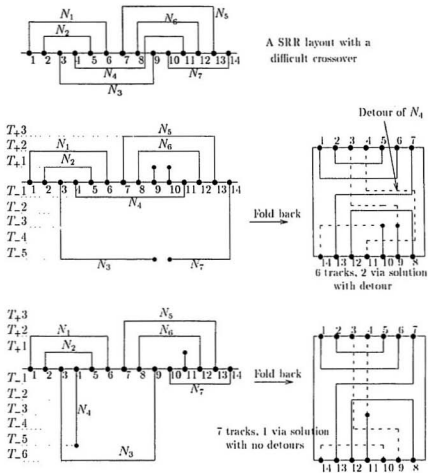


Figure 7.7: Handling a crossover on a straight net

put channel problem. Such a guaranteed routing without manual intervention is essential in a fully automated VLSI layout system.

For a fixed grid channel model the free grid point needed to align vias for some channel problems cannot be generated directly, since that column is already used by some other net and terminals are not movable. A scheme is needed which preserves a fixed terminal environment for the external world but allows a free column inside the channel domain. A new scheme called the *grid opening* technique is now proposed which is general, simple, and frees up the grid columns wherever needed. The grid opening technique can be visualised as a substitute for a detour to free up a grid point to align a via. The scheme is straightforward and does not need the complex process as required for detouring. The drawback of this technique is that it may extend the length of the channel.

The proposed technique uses the steps of *pseudo elongated channel formation* and *river routing*. Basically inside the routing domain another pseudo channel is formed whose terminals line up with the original channel problem to start with. Wherever there is a difficult to align crossover in the SRR, no attempt is made to handle it but the corresponding interval in the channel domain is widened to accommodate the via. For example, if there is a single crossover then the interval is expanded to twice the original interval. This allows a free intermediate column on which the via can be aligned. This increases the length of the pseudo channel as well as pushes all the terminals to the right of the expanded interval. However, the terminals of the original problem still remain intact. Now the problem of connecting the terminals of the original channel and that of the elongated pseudo channel can be solved by river routing. River routing is a special case of a channel in which the order of top terminals of the nets is the same as the order of their bottom terminals and every net is a two pin net. The overall approach is illustrated in Figure 7.8. It is to be noted that one grid expansion of the pseudo channel can accommodate two crossovers if they are at mirror intervals, which lie one below

another after the fold back. In general the increased length of the pseudo channel is equal to the number of crossovers. The top and bottom river routing problems can be solved in linear time using the existing river routing algorithms [Hsu86].

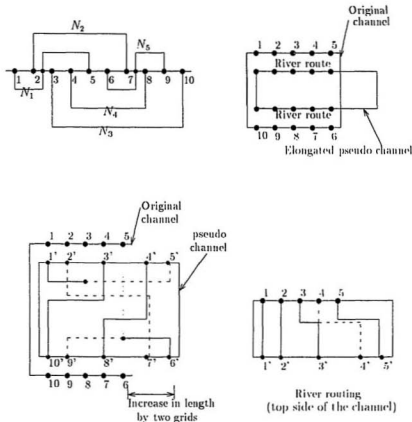


Figure 7.8: Grid opening technique to handle crossovers

The technique is general and is guaranteed to work for any type of crossover, because the nature of the crossover has no impact on the procedure. The basic problem of non-available columns for crossover alignment is solved by adding free columns when needed. However, always using grid opening is not recommended because not all crossovers need grid opening for alignment. For most

cases, there is a way of aligning the crossovers without splitting the grid. Further, using a 2-way overlap model, extra columns are not needed as often as in the case of the conventional HV-model, because loops in the Vertical Constraint Graph do not need a dogleg in the case of 2-way overlap model. If the grid opening technique is used indiscriminately the length of the pseudo channel increases which requires additional columns and hence more area. It is recommended that grid opening should be used as the last resort when the other simple techniques fail and the increase in area and extra work of river routing is justified.

### 7.3 Fold back step of the BT

The fold back step takes the layout produced by the crossover handler and maps the wire segments of the SRR layout onto the 2-layer channel. All segments in the upper street are assigned to one layer and all segments in the lower street are assigned to the opposite layer. The split segments of nets should contact through a via after the fold back. The fold back should be intelligent in the sense that it should do more than a naive translation of the SRR layout into a channel layout. The folding back operation should also exploit the features allowed by the channel model. For example, the channel width can usually be decreased by allowing overlaps, knock-knees and doglegging. Simply deforming the SRR layout and folding back retaining all the pseudo stretches of the SRR nets cause double runs in the channel. This increases the channel density thereby decreasing the quality of the routing solution and hence pseudo stretches should be avoided in the final channel layout. Figure 7.9 illustrates different ways of folding back an SRR layout. Obviously, allowing overlaps and knock-knees while folding back leads to a decrease channel width. Thus, the less constrained the routing model is, the more sophisticated the folding back needs to be in order to exploit all the features allowed by the routing model.

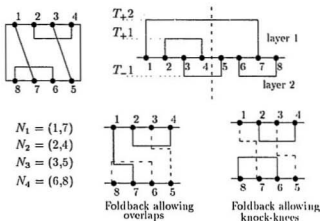


Figure 7.9: Different ways of folding back an SRR layout

Overlapping of nets also helps to reduce the channel width in a similar way. Further, overlapping of nets increases routability in some cases, especially channels with a loop in VCG which cannot be routed by the conventional HV model. However, long overlapping of nets needs to be avoided as it could lead to signal degradation and crosstalk due to capacitive coupling. In fact, overlap reduction and knock-knee maximisation can be attempted as post BT operations after obtaining the channel layout, in a way analogous to the constrained via minimisation of channels.

It is possible in some cases to reduce the channel width by a judicious introduction of doglegs. However, incorporating this process into the fold back step of BT is not easy because finding a good placement for the doglegs is non-trivial in the general case. An alternate way is to absorb the detours into the Single Row Routing step itself. That is, by allowing backward moves in the SRR, realisations with detours are produced which when translated into a channel would manifest as channel detours. Although [Ragh84] and [Tarn84] discuss SRR with backmoves, no algorithm has been reported yet to route an SRR problem with detours.

Finally, the fold back should be non-geometric in order to facilitate its implementation. Geometric fold back schemes which handle the layout information explicitly and operate on a segment by segment basis get unwieldy for large examples. Further, each SRR layout needs a different handling technique based on the nature and complexity of the crossover. One approach is to try out the simpler handling techniques and to use the grid open technique as the last resort. In realistic cases when there are multiple crossovers the interaction between them plays a role in deciding the order of handling and the technique to be chosen as well. In fact an algorithmic approach to BT to handle real life examples would be rather involved. A rule based expert system approach appears to be a good candidate, where the rules capture the most common layout patterns and give modified layouts for them. More research is needed in developing such an artificial intelligence based approach to crossover handling.

## Chapter 8

# Software development of an SRT based router

The goal of this chapter is to apply the results of the analysis carried out so far to the software development of an SRT based channel router. To start, the structure of the software for a general channel router based on the *abcd* type SRT is presented. Then, the special cases of a via free channel router and a Permutation channel router are discussed. Next, details of implementation highlighting the data structures and subroutines to be used are presented. The code has been tested on different bipartite examples and the results of some test cases are shown.

### 8.1 Structure of the software

A channel router based on the SRT approach should have the basic modules of the FT, the SRR solver and the BT in addition to the utilities such as plot routines. The FT would be the *abcd* type and can be performed by a trivial manipulation of the top and bottom net vectors of the channel. The SRR problem generated by



the FT is then routed by the techniques discussed earlier to get an SRR layout. The BT step which produces the channel realisation has to be sophisticated to fully exploit the allowed features (say knock-knee, overlap) of the chosen routing model to reduce the channel width. The SRR solving step is fairly independent of the channel model. The input channel could be a general or a special channel and these cases are discussed below.

### **8.1.1 General router**

The general channel router should be capable of routing any rectangular channel problem. Only 2-pin nets are considered, because any multi-terminal net can be decomposed into two pin nets and routed. The channel can be a permutation channel or a mixed channel and bipartite or non-bipartite. Based on the analysis in Chapter 5 and Chapter 6, it is clear that the box-procedure can route only a PBSRR, whereas the TARNG-MOD algorithm (using Tarng's ordering combined with 2-colouring order) can route both permutation and mixed channels. Further, the PNSRR problems can be routed by the box-procedure and the MNSRR problem should be broken into PNSRR problems and routed. The general channel router should have all these modules built in and should use them selectively depending upon the input problem. The structure of the software for such a general router is shown in Figure 8.1.

### **8.1.2 Special routers**

Two special types of channel routers can be designed using the analysis carried out so far on SRT based routing. The first is a bipartite channel router where the input problem has a via free solution and could be an PBSRR or an MBSRR problem which can be routed by the box-procedure and TARNG-MOD algorithms respec-

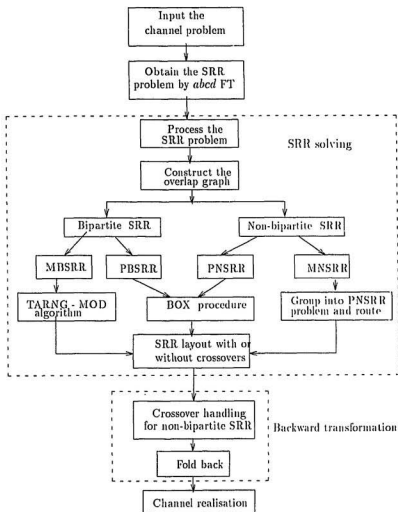


Figure 8.1: Software structure of a general SRT router

tively, to get the SRR solution. In this case the SRR layout need not be explicitly obtained and then folded back; the BT can be done directly by using simple algebraic relations between the level (track number) of the net in the conceptual SRR realisation and its level in the final channel. Of course, such relations should include the possible channel width reduction considerations. The second specialised router is a permutation channel router which is to route a PBSRR or a PNSRR problem. Here the box-procedure is used which would produce either a crossover free solution or a minimum crossover solution. If the problem is non-bipartite then crossover handling must be done before folding the channel back. Figure 8.2 depicts the software structure of a bipartite channel router and a permutation channel router based on the SRT approach.

## 8.2 Implementation details and results

This section gives details on the implementation in terms of the data structures to be used and subroutines that would be necessary.

- Data structures used

Nets and nodes are the basic entities of the router software and thus two basic data structures, form the database built and operated upon by different subroutines. The *net\_info* data structure is built to facilitate a quick answer to attributes related to nets, namely, net number, the left and right nodes of the net, zone to which net belongs, level of the net in the conceptual realisation, colour of the net, and so on. The *node\_info* data structure contains the information about each node of the SRR problem, namely, type of the node (Begin / End / Middle), net to which the node belongs, upper and lower node cut numbers, *etc.* Constants, like street congestion, number of nets, number of nodes, *etc.* are stored in the structure *cons\_str*.

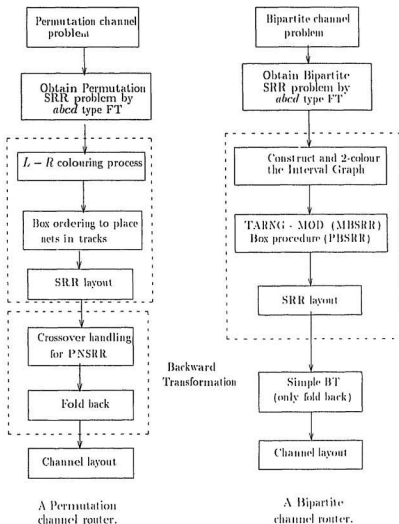


Figure 8.2: Software structure for special channel routers

- Subroutines used

The subroutines have been developed keeping in mind functionality, modularity and expandability. The basic set of subroutines which have been coded are listed below.

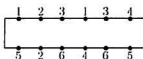
**Subroutine : Purpose**

<code>get_channel</code>	: Gets the channel information as top and bottom net vectors.
<code>abcd_FT</code>	: Forms the SRR problem from the top and bottom net vectors.
<code>process_SRR</code>	: Finds the node and net cut numbers of the SRR problem.
<code>construct_IG</code>	: Constructs the overlap graph and attempts bipartition.
<code>zone_allot</code>	: Allots nets to zones based on their total cut numbers.
<code>net_order</code>	: Uses residual cut numbers to order nets within a zone.
<code>track_assign</code>	: Uses Tarnig's order and 2-colouring information to place the nets on tracks to form a conceptual realisation.
<code>plot_srr</code>	: Converts the conceptual realisation into a physical realisation and creates a Latex plot file.
<code>easy_BT</code>	: Folds the conceptual SRR realisation into a channel.
<code>plot_channel</code>	: Creates a plot of the final channel layout, using dashed and solid lines for the two layers.

Apart from these subroutines, to handle the non-bipartite case, crossover handling by the easy align, split-align or grid opening techniques, as well as the subsequent fold back operations are necessary to form a full fledged router. The code has been successfully tested on different bipartite examples. Some examples of the routing solution produced by the router are shown in Figure 8.3 and Figure 8.4. It is to be noted that in Figure 8.4 although the SRR solution has a congestion of three while the optimal congestion is two, the final channel width is not increased. This is because the advantage of having fewer tracks in the SRR solution is lost, in many cases, in the extra tracks needed to align the crossovers. Thus, a solution with crossovers usually leads to no smaller channel width than a solution without

crossovers. In general, the SRT based router produces solutions of quality comparable to those produced by other routers. For the non-bipartite problems, the quality of the final channel solution depends both on the input problem's complexity and the efficacy of the Backward Transformation.

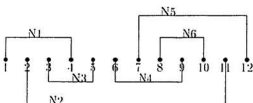
(A). Pictorial representation of the Input Channel.



(B). Equivalent SRR Problem using *abcd* FT.

Nets : 1 2 3 1 3 4 5 6 4 6 2 5  
Nodes : 1 2 3 4 5 6 7 8 9 10 11 12

(C) Realisation of *bipart1.srr*



(D). 2-layer Realisation of *bipart1.chan*

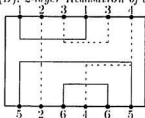
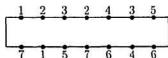


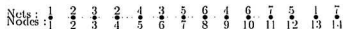
Figure 8.3: A channel problem routed by the implemented router

Plots Created by ROUTER.c on Sat Mar 3 23:04:06 1990

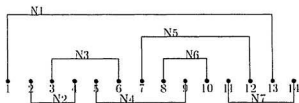
(A). Pictorial representation of the Input Channel.



(B). Equivalent SRR Problem using *abcd* FT.



(C) Realisation of *bipart3.srr*



(D). 2-layer Realisation of *bipart3.chan*

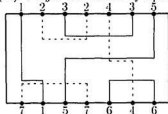


Figure 8.4: A channel problem routed by the implemented router



## Chapter 9

### Conclusions

This thesis analysed the Single Row Transformation approach to VLSI channel routing. The steps of the Forward Transformation, SRR solving and the Backward Transformation were discussed. The *abcd* FT-BT was chosen, for which the FT is straightforward and the BT, in the absence of crossovers, is simple. However, the presence of crossovers makes the BT highly case dependent and hence complex, leading to the necessity of minimum crossover routing of the SRR problem. Since the existing SRR algorithms were not designed for crossover minimisation, various algorithms for minimum crossover routing of the SRR problem were discussed. Since not all SRR problems are crossover free, crossover handling techniques are essential and the easy align and split align techniques were discussed, which however, fail for some difficult crossovers. A generalised handling technique called grid open, which is capable of handling any crossover was proposed to ensure 100% channel routability.

Since the general cases for both the channel and the single row routing problems are NP-complete, the conceptual simplicity of the SRT approach is not accompanied by a reduction in the computational complexity. In order to make

the approach viable only simple SRTs should be chosen. For the *abcd* SRT which is recommended, the FT takes  $O(n)$  time, where  $n$  is the total number of terminals in the routing problem. The BT in the absence of crossovers and detouring in the channel can be done in  $O(n \log n)$  time, by the simple algebraic relation between the track number in the conceptual SRR realisation and the track number in the final channel. Apart from its polynomial time complexity, bipartite channels have a 100% guaranteed routing solution, requiring in the worst case twice the SRR congestion. This clearly shows the SRT approach using the *abcd* type FT-BT is a good candidate for the special case of bipartite channels which can be routed without any vias.

The SRT approach is quite suitable for routing Permutation channels because the box procedure is able to route successfully the PBSRR problems as well as the PNSRR problems. However, in the case of PNSRR problems, the BT step has to incorporate suitable crossover collapsing as well as the subsequent crossover handling via easy align/split align or the grid open techniques. To route an MNSRR problem, groups need to be identified which can be done in  $O(n \log n)$  time [Gupt82], as the groups of the SRR problem can be interpreted as the maximal cliques of the overall interval graph  $G_I$  of the SRR problem [Sher89]. Then each of the PNSRR problems has to be routed and the solutions need to be merged; such merging, satisfying the net order within each group, however, is not possible in all cases. Thus, for the non-bipartite channels, the SRT approach is more complex and the solution could potentially have larger channel widths. Incorporating selective detouring improves routability as well as decreases the channel width for non-bipartite channels; however, at present, the best approach to incorporate such channel detours is not known.

More research is needed on the crossover handling step. Although the grid open technique is general and guaranteed to work, experience with many examples has shown that opening the grid is not needed and there is almost always

a way to align the vias to an existing grid. However, an algorithmic approach to specify such a solution is rather complex and hence a rule based system approach emerges as a good candidate. The rules have to capture the knowledge about the possible SRR layouts with representative crossover patterns and the corresponding solution after handling. An inference engine has to excite this knowledge base to extract the appropriate solution for the particular input problems. Such an artificial intelligence approach to crossover handling remains to be proven.

Additional work is needed on the multi-crossover handling techniques which take into consideration the interaction between crossovers while attempting the crossover handling. Further, some post BT operations are possible. For example, overlap reduction and knock knee maximisation could be formulated and studied in a manner analogous to the constrained via minimisation of channels. However, each of these problems is complex and it is uncertain whether the results of such post BT operations would be justified by improvements in the channel quality.

This thesis addressed only the problem of channel routing using the SRT approach. However, the SRT approach is applicable to any closed polygonal (bounded) routing region. The routing of switchbox problems, other complex shaped channels (*e.g.* L-shaped, X-shaped, T-shaped *etc.*) and circular channels (as in a pad frame router) are some interesting possibilities. This thesis assumed a 2-layer channel model; however, it would be quite interesting to extend the SRT approach to multi-layer schemes. The practicality of the approach for different design styles remains to be evaluated by integrating the SRT based router into an existing CAD tools environment and testing it on realistic designs.

There are a wealth of problems related to the SRT approach to VLSI routing and the results presented in this thesis form a basis on which future research could build.

# Bibliography

- [Aker67] Akers, S.B., *A modification of Lee's path connection algorithm*, IEEE Trans. Electron. Comput., Feb. 1967, pp. [97–98].
- [Bhat88] Bhattacharya, B.B., Deogun, J.S., Sherwani, N., *A Graph theoretic approach to single row routing problems*, Proc. IEEE Intl. Symp. on Ckt. & Systems, ISCAS 1988, pp [1437–1440].
- [Cies81] Ciesielski, M.J., Kinnen, E., *An optimum layer assignment for routing of ICs and PCBs*, Proc. 18<sup>th</sup> Design Automation Conf., June 1981, pp. [733–737].
- [Chan87] Chang, K.C., Du, H.C., *Efficient algorithms for the layer assignment problem*, IEEE Trans. on Computer Aided Design, Vol. CAD-6, No.1, Jan. 1987, pp [67–78].
- [Deut76] Deutsch, D.N., *A dogleg channel router*, Proc. 13<sup>th</sup> Design Automation Conf., June 1976, pp. [425–433].
- [Du87] Du, D.H., Ibarra, O.H., Naveda, J.F., *Single row routing with crossover bound*, IEEE Trans. on Computer Aided Design, Vol. CAD-6, No.2, March 1987, pp [190–201].
- [DuLi87] Du and Liu *Heuristic algorithms for single row routing*, IEEE Trans. on computers, Vol.C 36, No.3, March 1987, pp [312–320]
- [Even72] Even, S. Pnueli, A. *Permutation Graphs and Transitive Graphs*, Journal of ACM, Vol. 19, No. 3, July 72, pp [400–410].
- [Fost79] Foster J.C., *A Look-ahead Router for Multilayer printed wiring boards*, Proc. 16<sup>th</sup> Design Automation Conference, 1979, pp [486–493].
- [Gavr73] Gavril, F., *Algorithms for a maximum clique and a maximum independent set of a circle graph*, Networks, Vol. 3, 1973, pp [261–273]
- [Gare79] Garey M.R, Johnson D.S, *Computers and Intractability*. 1979.
- [Golu80] Golumbic M.C, *Algorithmic graph theory and perfect graphs*, Academic Press.

- [Gupt82] Gupta, U.I., Lee, D.T., Leung, Y.Y.T., *Efficient algorithms for Interval graphs and Circular arc graphs*, Networks, Vol. 12, 1982, pp [459–467].
- [Han84] Han, S., Sahni, S., *Single row routing in narrow streets*, IEEE Trans. on Computer Aided Design, vol. CAD-3, No. 3, July 1984, pp [235–241].
- [Han85] Han, S., Sahni, S., *A fast algorithm for single row routing*, Proc. 23rd Allerton Conf. on Communication, Control and Computers, 1985, pp [676–685].
- [Hash71] Hashimoto, A., Stevens, S., *Wire routing by optimising channel assignment within large apertures*, Proc. 8<sup>th</sup> Design Automation Conf., 1971, pp [155–169].
- [High69] Hightower, D., *A solution to line-routing problems on the continuous plane*, Proc. Design Automation Workshop, 1969.
- [Hsu83] Hsu Chi-Ping, *Minimum via Topological routing*, IEEE Trans. on Computer Aided Design, vol. CAD-2, No. 5, Oct. 1983, pp [235–246].
- [Hsu86] Hsu Chi-Ping, *Signal Routing in Integrated Circuit Layout*, UMI Press, 1986.
- [Joob86] Joobhani, R., *An artificial intelligence approach to VLSI routing*, Kluwer Academic Publishers, 1986.
- [Kaji80] Kajitani, Y., *On via hole minimization of routing in 2-layer boards*, Proc. IEEE Int. Conf. on Circuits and Computers, ICC, June 80, pp [295–298].
- [Karp72] Karp, R.M., *Reducibility among combinatorial problems*, Complexity of Computer Communications, Eds., Miller, R.E., Thatcher, J.W., Plenum Press, 1972, pp [85–104].
- [Kuh79] Kuh, E.S., *On optimum single row routing*, Proc. IEEE Trans. on Ckts & Systems, Vol. CAS-26, No. 6, June 79, pp [361–368].
- [Lee61] Lee, C.Y., *An algorithm for path connection and its applications*, IRE Trans. Electron. Comput., Sept. 1961, pp. [346–365].
- [Lloy89] Lloyd, E.L., *A fast algorithm for finding interlocking sets*, Information Processing Letters, Vol 32, (1989), 3<sup>rd</sup> July 89, pp [47–50].
- [Ragh80] Raghavan, R., Sahni S., *Single-row routing*, Tech Report 80-20, Dept. of Computer Science, University of Minnesota, 1980.
- [Ragh81] Raghavan, R., Sahni, S., *Optimum single row router*, Proc. 19th Design Automation Conference, 1982, pp [38–45].
- [Ragh83] Raghavan, R., Sahni, S., *Single row routing*, IEEE Trans. on computers, Vol. C-32, No.3, march 1983, pp [209–220].

- [Ragh84] Raghavan, R., Sahni, S., *The complexity of Single Row Routing*, IEEE Trans. on ckt's & systems, Vol. CAS-31, No. 5, May 1984, pp [462-472].
- [Rim89] Rim, C.S., Kashiwabara, T., Nakajima, K., *Exact algorithms for multi-layer Topological via minimization*, IEEE Trans on CAD, Vol. CAD-5, no. 11, Nov 89, pp [1165-1173].
- [Rive82] Rivest, R.L., Fiduccia, C.M., *A greedy channel router*, Proc. 19<sup>th</sup> Design Automation Conf., June 1982, pp [418-424].
- [Rubi74] Rubin, F., *The Lee connection algorithm*, IEEE Trans.on Comp., Vol. C-23, 1974, pp. [907-914].
- [Sadw84] Sadwoska, M., *An unconstrained Topological via minimization problem for two-layer routing*, IEEE Trans on CAD, Vol. CAD-3, no. 3, July 84, pp [184-190].
- [Sarr87] Sarrafzadeh, M., *Channel routing in the knock knee mode is NP-complete*, IEEE Trans on CAD, Vol. CAD-6, no. 4, July 87, pp [503-506].
- [Sarr89] Sarrafzadeh, M., *A new approach to Topological via minimization*, IEEE Trans on CAD, Vol. CAD-5, no. 8, Aug. 89, pp [890-900].
- [Saxe89] Saxena, S., Prasad, V.C., *On Single Row Routing*, IEEE Trans. on ckt's & systems. Vol. 36, No. 7, July 89, pp [1029-1032].
- [Sher89] Sherwani, N., Deogun, J.S., *A new heuristic for single row routing problems*, Proc. 26<sup>th</sup> Design Automation Conference, 1989, pp [167-172].
- [So74] So, H.P., *Some theoretical results on routing multilayer boards*, Proc. IEEE Intl. Symp on Circuit & Systems, 1974, pp. [296-303].
- [Supo85] Supowit. K.J., *Decomposing a set of points into chains, with applications to permutation and circle graphs*, Information Proc. Letters, Vol. 21, 1985, pp [249-252].
- [Supo87] Supowit. K.J., *Finding a maximum planar subset of nets in a channel*, IEEE Trans on CAD, vol 6, no. 1, Jan. 87, pp [93-94].
- [Szym85] Szymanski, T.G., *Dogleg channel routing is NP-complete*, IEEE Trans on CAD, vol 4, no. 1, Jan. 85, pp [31-41].
- [Tarn84] Tarn, T.T.K., Sadwoska, M., Kuh, E.S., *An efficient single row routing algorithm*, IEEE Trans. on CAD, Vol. CAD-3, no. 3, July 84, pp [178-183].
- [Ting76] Ting, B.S., *The multilayer routing problem: Algorithms and necessary and sufficient conditions for the single row single layer case*, IEEE Trans. on ckt's & systems, vol. CAS-23, No. 12, December 1976, pp. [768-777].

- [Tsui81] Tsui, R.Y, Smith, R.J., *A high density multi layer PCB router based on necessary and sufficient conditions for single row routing*, Proc. 18<sup>th</sup> Design Automation Conference, 1981, pp [372-381].
- [Tsuk80] Tsukiyama, S., Kuh, E.S., *Double row planar routing and permutation layout*, Univ. of California at Berkeley, Elect. Res. Lab. Memo, UCB / ERL / M80 / 46, July 80.
- [Veng90] Venguswamy, K, *Via free routing of Permutation Channels*, accepted at the Canadian Conf. on VLSI, CCVLSI-90, October 90, Ottawa.
- [Wong88] Wong J.S.L and Kwok *A single row transformation technique*, IEEE Design and Test, vol.5, no.1, Feb 1988, pp [43-47].
- [Xion88] Xiong, X.M., *A new algorithm for topological routing and via minimization*, Proc. Int. conf. on Computer Aided Design, ICCAD, Nov. 1988.
- [Xion89] Xiong, X.M., Kuh, E.S., *An unified approach to the via minimization problems*, Proc. IEEE Trans. on ckt & systems, Vol. CAS-36, no. 2, Feb. 89, pp [190-203].
- [Yosh82] Yoshimura, T., Kuh, E.S., *Efficient algorithms for channel routing*, IEEE Trans. on CAD of ICAS, Vol. CAD-1, no. 1, Jan. 1982, pp. [25-35].







