1

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

PIONEERS IN METHYLENE BLUE TESTING SINCE 1974

10800 COUNTRY ROAD 5 BLUMENVILLE, MINN. 55007 USA
TEL. 612 426 7887 FAX: 612 426 7887 T.L.X. 510000HDS

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file - Votre référence

Our file - Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

A Topological Single-Layer Routing Algorithm and Its Application to Leaf Cell Synthesis

by

Carol R. Bachelu, B.A.Sc.

**A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of**

Master of Engineering

**Ottawa-Carleton Institute for Electrical Engineering
Department of Electronics
Faculty of Engineering
Carleton University
Ottawa, Ontario
September, 1992**

© 1992, Carol R. Bachelu

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Vous les avez obtenus

Vous les avez obtenus

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.

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.

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 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 0-315-79781-9

Canada

The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis

**“A Topological Single-Layer Routing Algorithm
and Its Application to Leaf Cell Synthesis”**

submitted by Carol R. Bachelu, B.A.Sc.
in partial fulfilment of the requirements for
the degree of Master of Engineering

**Professor M.C. Lefebvre
Thesis Supervisor**

**Professor J.S. Wight
Chairman,
Department of Electronics**

**Carleton University
September 21, 1992**

Abstract

A new solution is proposed for the leaf cell routing problem. The salient feature of this approach is that it takes a global view of the routing problem by considering all the nets concurrently. This is in contrast with the local optimization performed by most existing algorithms that operate on a net by net basis. Based on a graph-theoretic abstraction of the layout space, a topological ordering of wires is created and used as a guide in the assignment of final layout geometries. This solution also considers connections on the local interconnect layer which is available in advanced CMOS technologies.

Acknowledgements

Without the efforts and assistance of two individuals, this thesis would not have been possible. I wish to convey my immense gratitude to my supervisor, Dr. Martin Lefebvre, for presenting to me the initial conception of a topological router upon which this thesis is based. His continued assistance with and interest in the development of the prototype software have been invaluable, his comments and suggestions instrumental in ensuring the consistency and coherence of the written thesis. I also wish to thank David Skoll for bringing to my attention the elegant algorithm for detecting crossing routes in a graph, and for his work on the branch and bound algorithms which perform the crucial function of selecting the solution set of metal 1 connections. Finally, thanks are extended to the faculty, staff, and my fellow graduate students of the Department of Electronics for their support.

I wish to acknowledge the following sources of financial support: the Natural Sciences and Engineering Research Council (N.S.E.R.C) through a Postgraduate Scholarship; Bell-Northern Research Limited (B.N.R), Ottawa, through a Postgraduate Scholarship; and the Department of Electronics, Carleton University. The prototype software for the topological router was developed on a network of SUN SPARCstations provided by the Canadian Microelectronics Corporation (C.M.C).

Table of Contents

Abstract	i
Acknowledgements	ii
Table of Contents	iii
List of Tables	viii
List of Figures	ix
Chapter 1. Introduction	1
1.1 Perspective	1
1.2 Overview of Leaf Cell Routing Approaches	1
1.3 Problem Formulation	4
1.4 Thesis Objectives	5
1.5 Organization of the Thesis	5
Chapter 2. Topological Single-Layer Routing Algorithm	7
2.1 Basic Concepts	7
2.2 Layout Abstraction	10
2.3 Detection and Removal of Equivalent Paths	16
2.4 Detection of Crossing Paths	18
2.5 Optimal Solution Selection	19
2.6 Reduction Heuristic	21
Chapter 3. Application of the Topological Single-Layer Routing Algorithm to Leaf Cell Routing	26
3.1 Reduction of the Leaf Cell Routing Problem to a Single Layer Routing Problem	27

3.1.1 Diffusion Connections	29
3.1.2 Poly Connections	30
3.1.3 LI Connections	33
3.2 Overview of the Algorithm	38
3.3 Adjacency Graph Structure for the First Pass	42
3.3.1 Region Elimination	46
3.3.2 Pin Elimination	46
3.3.3 Region Merging	46
3.3.4 Adjacency Graph	50
3.4 Determining the Minimum Set of Two-Point Connections for the Metal1 Nets	51
3.4.1 Sub-nets	51
3.4.2 OR-Groups	51
3.4.3 Minimum Set of Sub-nets for a Net	53
3.4.4 Using Virtual Pins to Split Long Sub-nets	54
3.5 Finding All Paths For A Sub-net	55
3.5.1 Restricting the Boundaries of the Search Space for the Paths of a Sub- net	56
3.5.1 Preventing the Generation of Tightly Zigzagging Paths	59
3.5.2 Early Detection and Termination of Invalid Paths	60
3.5.3 Termination of Paths at Diffusion Regions	60
3.5.4 Storing All the Unique Paths of All Nets	62
3.6 Detection and Removal of Equivalent Paths	62
3.7 Reduction of the Path Set Prior to Solution Selection	64
3.7.1 Assigning a Cost to Each First-Graph Path	64
3.7.2 Recording Crossing Paths	64
3.7.3 Reducing the Set of First-Graph Paths	67
3.7.4 Eliminating Virtual Pins to Recombine Long Sub-nets	68
3.8 Selecting the Solution Set of First-Graph Paths	71
3.8.1 Branch and Bound Selection	71
3.8.2 Determining Available Tracks for the Solution	74
3.9 Adjacency Graph Structure for the Second Pass	75
3.9.1 Region Elimination	78
3.9.2 Adjacency Graph	83

3.10 Finding All Paths on the Second Graph for Each of the First-Graph Solution Paths	83
3.11 Detection and Removal of Equivalent Paths	86
3.12 Applying the Reduction Heuristic to the Set of Second Graph Paths	87
3.13 Selecting a Final Solution Set of Second-Graph Paths	87
3.14 Mapping the Final Solution Into Symbolic Layout	87
Chapter 4. TSLR with an Example Leaf Cell Circuit	94
4.1 The Transmission Gate XNOR Circuit	94
4.2 Reduction to a Metall Routing Problem	96
4.3 The Minimum Set of Two-Point Metall Connections	97
4.4 The Set of All First-Graph Paths for the Sub-nets	98
4.5 Selection of a Solution Set of First-Graph Paths	101
4.6 The Set of All Second-Graph Paths for the Sub-nets	102
4.7 Selection of the Final Solution Set of Paths	105
4.8 Symbolic Layout Representation of the Solution	106
Chapter 5. Experimental Results	109
5.1 Test Circuits	110
5.2 First Pass Adjacency Graph	112
5.2.1 Case 1: LI is used for routing, but no routing is permitted in the top and bottom track areas.	112
5.2.2 Case 2: LI is used for routing, and routing is permitted in the top and bottom track areas.	117
5.2.3 Case 3: LI is not used for routing, and no routing is permitted in the top and bottom track areas.	124
5.2.4 Case 4: LI is not used for routing, but routing is permitted in the top and bottom track areas.	125
5.3 Path Generation for the First Pass	127
5.4 Reduction Algorithm	128
5.4.1 Calculating Costs for the Paths	128

5.4.2 Reducing the Path Sets	128
5.4.3 Eliminating Virtual Pins	129
5.5 Branch and Bound Solution for the First Pass	130
5.6 Second Pass Adjacency Graph	133
5.7 Path Generation for the Second Pass	137
5.8 Branch and Bound Solution for the Second Pass	138
Chapter 6. Conclusions	140
6.1 Summary	140
6.2 Contributions	141
6.3 Future Work	141
6.3.1 Virtual Pins	141
6.3.2 Mapping to Symbolic Layout	142
6.3.3 I/O Ports	142
6.3.4 Effects of Transistor Sizing	142
6.3.5 Reduction Algorithm for the Second Pass	143
6.3.6 Reduction Algorithm	143
6.3.7 Branch and Bound Solution Search	144
6.3.8 Constrained Layout	144
Appendices	145
A. Glossary of Terms	145
B. Examples of the Cases of Crossing Paths	147
B.1 Crossings Paths With a Single Region in the Common Section	147
B.2 Crossings Paths With at Least Two Regions in the Common Section	150
C. Test Circuit Sub-nets (by or-group)	152
C.1 Case 1: LI is used for routing, but no routing is permitted in the top and bottom track areas.	152
C.2 Case 2: LI is used for routing, and routing is permitted in the top and bottom track areas.	154
C.3 Case 3: LI is not used for routing, and no routing is permitted in the top and bottom track areas.	156
C.4 Case 4: LI is not used for routing, but routing is permitted in the top and bottom track areas.	156
D. Test Circuit Long Sub-nets	157

D.1 Non-dual full adder: routing is not permitted in the top and bottom track areas	157
D.2 JK flip flop: routing is not permitted in the top and bottom track areas	157
D.3 Master-slave D flip flop: routing is not permitted in the top and bottom track areas	158
D.4 Master-slave D flip flop: routing is permitted in the top and bottom track areas	158
References	161

List of Tables

Table 2.1	Paths for arc sets A_1 and A_2	.12
Table 2.2	Paths for arcs in the reduced set Q_1	.15
Table 2.3	Arc sets A_1 and A_2	.21
Table 3.1	Region correspondence between first and second graphs	86
Table 4.1	Region correspondance between first and second pass	103
Table 4.2	Horizontal and vertical solution path segments	106
Table 4.3	Pin coordinates	107
Table 5.1	Case 1 Graph Statistics	.6
Table 5.2	Case 1 Sub-net Statistics.	117
Table 5.3	Case 2 Graph Statistics	118
Table 5.4	Case 2 Sub-net Statistics.	120
Table 5.5	Case 3 Graph Statistics	124
Table 5.6	Case 3 Sub-net Statistics.	125
Table 5.7	Case 4 Graph Statistics	125
Table 5.8	Case 4 Sub-net Statistics.	126
Table 5.9	First Pass Path Statistics	127
Table 5.10	Cost Statistics for First Pass Paths	128
Table 5.11	Reduction Heuristic Statistics.	129
Table 5.12	Long Net Statistics	129
Table 5.13	First Pass Branch and Bound Statistics	130
Table 5.14	Second Pass Graph Statistics	137
Table 5.15	Second Pass Path Statistics.	137
Table 5.16	Second Pass Branch and Bound Statistics	138

List of Figures

Figure 1.1	Layout space with only a center routing channel	2
Figure 1.2	Layout space with minor routing regions	3
Figure 2.1	Nets N_1 and N_2	7
Figure 2.2	Arc set A_1	8
Figure 2.3	Arc set A_2	9
Figure 2.4	Nets N_1 and N_2 with boundaries and regions imposed	11
Figure 2.5	Adjacency graph for nets N_1 and N_2	12
Figure 2.6	Introduction of the obstacle, o_1	13
Figure 2.7	Equivalent and non-equivalent paths	16
Figure 2.8	Example adjacency graph	17
Figure 2.9	Example paths k_1 and k_2	17
Figure 2.10	Crossing paths.	18
Figure 2.11	Non-crossing paths	19
Figure 3.1	Topological single-layer routing as part of the leaf cell synthesis problem	26
Figure 3.2	Five horizontal areas of the layout space	28
Figure 3.3	Diffusion connection in neighboring transistors.	30
Figure 3.4	Power and ground connections in diffusion; common gate connection in poly	30
Figure 3.5	Poly connections: a) neighboring common gates, b) neighboring common p or n gate, c) neighboring common p gates or n gates, d) folded transistor gate	32
Figure 3.6	Common gate connection across a diffusion break	33
Figure 3.7	Neighboring common gate connection when one pair is missing a transistor	33
Figure 3.8	Common p to n diffusion connection in LI.	34
Figure 3.9	LI connections: a) common drain to common gate, b) common drain to gate, c) drain to gate	36

Figure 3.10	LI connections: a) common gate to common source, b) common gate to source, c) gate to source	.37
Figure 3.11	LI connection of neighboring diffusions of opposite type	.38
Figure 3.12	LI connection of equipotential diffusion terminals in a folded transistor	.38
Figure 3.13	Process flow of the topological single-layer leaf cell routing algorithm.	.41
Figure 3.14	Horizontal and vertical slicing imposed on the layout space	.42
Figure 3.15	Initial first graph region set for a transistor pair	.43
Figure 3.16	Region merging due to pin elimination	.47
Figure 3.17	Example region map	.48
Figure 3.18	Region merging due to shared diffusion.	.48
Figure 3.19	Region merging due shared diffusions of both types	.49
Figure 3.20	Region merging due to a missing n transistor.	.49
Figure 3.21	Region merging due to a missing p transistor.	.50
Figure 3.22	Example adjacency graph	.55
Figure 3.23	Checking for pins between regions in different columns	.63
Figure 3.24	Checking for pins between regions in the same column	.63
Figure 3.25	Non-overlapping paths	.65
Figure 3.26	Using extended paths to check for a crossing.	.67
Figure 3.27	Merging long nets: a) direct merge, b) modified direct merge	.69
Figure 3.27	cont ... c) merge with open loop removal, d) merge with closed loop removal	.70
Figure 3.28	Example net N_1	.73
Figure 3.29	Paths occupying the same horizontal segment	.74
Figure 3.30	Paths which cross	.74
Figure 3.31	Initial second graph region set for a transistor pair	.77
Figure 3.32	Example second pass region map	.79
Figure 3.33	Region merging due to shared diffusion.	.80
Figure 3.34	Complete elimination of the spacer regions	.81
Figure 3.35	Region merging due to a missing n transistor.	.82
Figure 3.36	Region merging due to a missing p transistor.	.83

Figure 3.37	Example first graph to second graph translation.	85
Figure 3.38	Tracks and routing columns imposed on an example second-pass region map.	90
Figure 3.39	Horizontal passage through a region	91
Figure 3.40	Vertical passage through a region	91
Figure 3.41	90 degree bends in a region	92
Figure 3.42	Changing tracks in a region	92
Figure 3.43	Changing routing columns in a region	93
Figure 4.1	Schematic diagram for the transmission gate XNOR circuit	94
Figure 4.2	Transistor placement	95
Figure 4.3	Region map for the first pass	95
Figure 4.4	Adjacency graph for the first pass	96
Figure 4.5	Region map for the second pass	102
Figure 4.6	Adjacency graph for the second pass	103
Figure 4.7	Imposition of tracks and routing columns on the layout space . . .	106
Figure 4.8	Metal1 solution wires	107
Figure 4.9	Compacted plot of the TSLR solution	108
Figure 5.1	Leaf cell synthesis: flow of operation.	109
Figure 5.2	Transmission gate XNOR placement	110
Figure 5.3	Non-dual full adder placement	111
Figure 5.4	JK flip flop placement.	111
Figure 5.5	Master-slave D flip flop placement.	111
Figure 5.6	Case 1 region map and adjacency graph for the transmission gate XNOR.	112
Figure 5.7	Case 1 region map and adjacency graph for the non-dual full adder	113
Figure 5.8	Case 1 region map and adjacency graph for the JK flip flop	114
Figure 5.9	Case 1 region map and adjacency graph for the master-slave D flip flop	115
Figure 5.10	Case 2 region map and adjacency graph for the transmission gate XNOR.	118

Figure 5.11	Case 2 region map and adjacency graph for the non-dual full adder	121
Figure 5.12	Case 2 region map and adjacency graph for the JK flip flop	122
Figure 5.13	Case 2 region map and adjacency graph for the master-slave D flip flop	123
Figure 5.14	Case 3 region map and adjacency graph for the transmission gate XNOR.	124
Figure 5.15	Case 4 region map and adjacency graph for the transmission gate XNOR.	126
Figure 5.16	First pass solution for XNOR w/o LI, no routing in top and bottom areas	131
Figure 5.17	First pass solution for XNOR w/o LI, routing in top and bottom areas	131
Figure 5.18	First pass solution for XNOR w/ LI, no routing in top and bottom areas	132
Figure 5.19	First pass solution for XNOR w/ LI, routing in top and bottom areas	132
Figure 5.20	Second pass region map and adjacency graph for XNOR w/o LI, no routing in top and bottom areas	133
Figure 5.21	Second pass region map and adjacency graph for XNOR w/o LI, routing in top and bottom areas	134
Figure 5.22	Second pass region map and adjacency graph for XNOR w/ LI, no routing in top and bottom areas	135
Figure 5.23	Second pass region map and adjacency graph for XNOR w/ LI, routing in top and bottom areas.	136
Figure 5.24	Plot of the compacted layout for XNOR, no routing in top and bottom area	139
Figure 5.25	Plot of the compacted layout for XNOR, routing in top and bottom areas	139

Chapter 1: Introduction

1.1 Perspective

Leaf cell synthesis consists of automatically generating layout geometries for transistor level structures normally contained in a component library. Arguments that have traditionally been brought forward to promote leaf cell generators include process independence, design flexibility, and the ability to customize individual cells to their desired environment. Significant qualitative differences between current sub-micron CMOS technologies processes make it very difficult to adapt cell designs from one technology to the next; in many cases, a complete redesign of the component library is required. When libraries are extensive, this can amount to a great deal of effort. With a component library consisting of leaf cell generators, only the section dealing with the actual mapping of the solution to a specific technology would need to be modified.

The leaf cell synthesis problem can be separated into three sub-problems: placement, routing, and compaction. In this thesis, a new solution is proposed for the leaf cell routing problem. The salient feature of the new approach is that it takes a global view of the routing problem as opposed to the local optimization performed by most existing algorithms.

1.2 Overview of Leaf Cell Routing Approaches

The line-of-diffusion style of layout for leaf cell circuits is the basis for most of the leaf cell routing algorithms proposed to date. This layout style arranges the transistors of the leaf cell circuit in two parallel, horizontal rows between the power and ground wires which run horizontally across the length of the cell at a fixed pitch. Usually, all p-type

transistors are placed in the upper row, and all n-type transistors are placed in the lower row. There is always an area of space in the centre of the cell between the parallel transistor rows due to physical minimum spacing requirements between the two types of diffusion.

Figure 1.1 Layout space with only a center routing channel

Many algorithms for leaf cell routing [1-4] are similar to existing channel routing algorithms used for inter-cell routing. These algorithms model the above layout space as parallel horizontal channels, (one for the row of p-type transistors, one for the row of n-type transistors and one for the space in the centre between the transistor rows) in which connections are made along horizontal routing tracks in one or more routing layers, usually metal and polysilicon (see Figure 1.1). The algorithm described in [1] is a modified channel router. It completes the majority of circuit connections in the centre channel, and uses the routing tracks in the two transistor channels as optional rows for routing. Similarly, the algorithm in [2] limits the number and type of connections completed in the transistor channels; the majority of connections are completed in the centre by a channel router with the restriction that vertical connections are implemented in polysilicon only. Alternatively, in the algorithms of [3,4], the majority of the circuit connections are completed in

the transistor channels. The centre channel is used mainly for connections between the two rows of transistors. Such connections are usually restricted to vertical wires.

The algorithms in [5,6,7] utilize an expanded model of the layout space which includes two additional horizontal areas between the power wire and the row of p-type transistors, and between the row of n-type transistors and the ground wire (see Figure 1.2). Like the algorithms of [1,2], the majority of connections are completed in the centre routing area (between the two rows of transistors). Certain connections are implemented in the other four routing areas to ease congestion in the centre as much as possible.

Figure 1.2 Layout space with minor routing regions

The algorithm in [8] differs from the previous algorithms in two ways: a) it uses a modified version of the line-of-diffusion style of layout which permits more than one row of transistors of each type, and b) it does not model the layout space in terms of horizontal routing channels. Instead, the algorithm views the entire layout space as a grid of locations. The transistors are simply obstacles at various locations on this grid. The algorithm

employs a maze router (with an A* search for back-tracking during the rip-up and re-route phase) which is permitted to route a connection anywhere within the space and in any direction as long as it avoids all obstacles. Pre-routed connections are also viewed as obstacles.

1.3 Problem Formulation

The leaf-cell router presented in this thesis builds on the approach in [8] of modeling the entire cell area as available routing area. However, instead of imposing a strict grid structure onto the space, it is divided into rectangular regions which are not necessarily the same size. Each of these regions is then assigned to a vertex in an adjacency graph. There is an edge between any two vertices in the graph whose corresponding regions are physically adjacent in the layout space. A connection between points in the circuit can then be described as an ordered series of these vertices. A solution for the circuit consists of a set of these "paths" through the adjacency graph, one for each connection in the circuit.

Of the layers available for routing connections in leaf cells, only the metal layers may be used to route connections throughout the entire layout space. Since the diffusion, polysilicon, and local interconnect layers have various restrictions on their use, as many of the circuit connections as possible are implemented in these layers. The router in this thesis uses only the first level of metal available for routing. Therefore, once all possible connections have been implemented in other routing layers, what remains is a single-layer routing problem.

The concept which differentiates this new leaf cell router from other leaf cell routing techniques is that the conventional practise of selecting and routing connections one at a time until all are complete is discarded, and instead, the adjacency graph is used to generate all practical paths for all connections in the circuit. Each of these paths is then con-

sidered in conjunction with every other path to determine the subset of paths which form the final solution.

One advantage of such a topological approach is that the rip-up and re-route problems inherent to the maze-router approach in [8] can be avoided because all the paths in the solution are known before wires are assigned to exact vertical and horizontal positions in the space. In addition, because the rectangular regions covering the layout space are not required to be the same size, circuits where the transistors have been sized can be easily handled.

1.4 Thesis Objectives

There are three objectives which drive this thesis. The first is to explore the feasibility of using a topological approach to leaf cell routing to produce acceptable layouts for leaf cell circuits. The second objective is to develop data representation methods and algorithms for the key components of a possible leaf cell router based on this approach. The final objective is to explore other necessary but secondary aspects of the problem such as mappings to symbolic layout, and methods for introducing the local interconnect layer in leaf cell synthesis. This last aspect builds on the work presented in [9] which showed that the local interconnect (LI) layer in full CMOS technologies may be used to reduce cell area and increase routing flexibility.

1.5 Organization of the Thesis

The remainder of the thesis is organized as follows. In Chapter 2, the proposed topological routing algorithm outlined in Section 1.3 is formally presented. Chapter 3 describes the application of this algorithm to the leaf cell routing problem. Chapter 4 contains a detailed example of the operation of this algorithm on a leaf cell circuit. In Chapter 5, the experimental results obtained with a prototype implementation of the topo-

logical routing algorithm for leaf cell circuits are presented. Finally, in Chapter 6, a set of conclusions about the new topological routing algorithm are stated, and some recommendations for future work are made.

Chapter 2: Topological Single-Layer Routing Algorithm

2.1 Basic Concepts

The basic routing problem can be defined in the following manner. Given a set of points in space and a set of rules which govern the connectivity relationships amongst the points, find a path through the space for each connection such that all connections can exist simultaneously. For example, let T be the set of nets $\{N_1, N_2\}$ where N_1 consists of the points $\{a_1, a_2, a_3\}$, and N_2 consists of the points $\{b_1, b_2, b_3\}$. The points are arranged in space as shown in Figure 2.1. In this case, let the connectivity be described by the following simple rule:

Connectivity Rule: All points of a net must be connected; points associated with different nets must not be connected.

Figure 2.1 nets N_1 and N_2

Working on the assumptions that a) there are no obstacles within the space (other than the points themselves), and b) there are no bounds on the space, then the set of arcs

$\Lambda_1 = \{r_1 r_2 \dots r_9\}$ shown in Figure 2.2 completely describes all possible two-point connections amongst the points of net N_1 .

Figure 2.2 arc set Λ_1

A similar set of nine arcs, $\Lambda_2 = \{p_1 p_2 \dots p_9\}$, exists for the points of net N_2 (see Figure 2.3). Note that to completely connect the points of net N_1 , it is only necessary to use one arc from any two of the following three connection sets:

$$(a_1 a_2): \{r_1 r_2 r_3\}$$

$$(a_1 a_3): \{r_4 r_5 r_6\}$$

$$(a_2 a_3): \{r_7 r_8 r_9\}$$

For example, if arcs r_1 and r_7 are used, arcs $r_2 - r_6$, and r_8, r_9 are redundant. The same holds true for the points of net N_2 .

Figure 2.3 arc set A_2

If P is defined to be the set of all possible solutions, where a solution consists of four arcs (two arcs from any two different connection sets for net N_1 and two arcs from any two different connection sets for net N_2), then the total number of possible solutions, is $(C^3_1 * C^6_1 + C^3_1 * C^3_1)^2 = (18 + 9)^2 = 729$.¹

If there is only one layer on which all the arcs must exist, then the connectivity rule is satisfied as long as the two selected arcs for net N_1 do not cross the two selected arcs for net N_2 . If the set of all valid solutions, V , is defined to be the subset of P for which no four-arc set violates the connectivity rule, then V is the following:

$$V = \{(r_1 r_4 p_1 p_7) (r_1 r_4 p_1 p_6) (r_1 r_4 p_3 p_7) (r_1 r_4 p_3 p_6) (r_1 r_4 p_6 p_7) (r_1 r_5 p_2 p_4) \\ (r_1 r_5 p_2 p_7) (r_1 r_5 p_4 p_7) (r_1 r_6 p_2 p_4) (r_1 r_7 p_1 p_4) (r_1 r_7 p_1 p_9) (r_1 r_7 p_3 p_4) \\ (r_1 r_7 p_3 p_9) (r_1 r_8 p_2 p_4) (r_1 r_8 p_2 p_7) (r_1 r_8 p_4 p_7) (r_1 r_9 p_2 p_7) (r_2 r_4 p_1 p_5) \\ (r_2 r_4 p_1 p_8) (r_2 r_4 p_3 p_5) (r_2 r_4 p_3 p_8) (r_2 r_5 p_2 p_8) (r_2 r_7 p_1 p_5) (r_2 r_7 p_1 p_8)\}$$

1. $C^m_n = m! / (m-n)!$

(r₂ r₇ p₃ p₅) (r₂ r₇ p₃ p₈) (r₂ r₈ p₂ p₅) (r₃ r₄ p₁ p₇) (r₃ r₅ p₂ p₄) (r₃ r₅ p₂ p₇)
 (r₃ r₅ p₄ p₇) (r₃ r₆ p₁ p₄) (r₃ r₇ p₁ p₄) (r₃ r₈ p₂ p₄) (r₃ r₈ p₂ p₇) (r₃ r₈ p₄ p₇)
 (r₃ r₉ p₁ p₇) (r₄ r₇ p₁ p₅) (r₄ r₇ p₁ p₈) (r₄ r₇ p₃ p₅) (r₄ r₇ p₃ p₈) (r₄ r₈ p₅ p₇)
 (r₄ r₉ p₁ p₇) (r₅ r₇ p₄ p₈) (r₅ r₈ p₂ p₄) (r₅ r₈ p₂ p₇) (r₅ r₈ p₄ p₇) (r₆ r₇ p₁ p₄)
 (p₁ p₄ r₁ r₇) (p₁ p₄ r₁ r₆) (p₁ p₄ r₃ r₇) (p₁ p₄ r₃ r₆) (p₁ p₄ r₆ r₇) (p₁ p₅ r₂ r₄)
 (p₁ p₅ r₂ r₇) (p₁ p₅ r₄ r₇) (p₁ p₆ r₂ r₄) (p₁ p₇ r₁ r₄) (p₁ p₇ r₁ r₉) (p₁ p₇ r₃ r₄)
 (p₁ p₇ r₃ r₉) (p₁ p₈ r₂ r₄) (p₁ p₈ r₂ r₇) (p₁ p₈ r₄ r₇) (p₁ p₉ r₂ r₇) (p₂ p₄ r₁ r₅)
 (p₂ p₄ r₁ r₈) (p₂ p₄ r₃ r₅) (p₂ p₄ r₃ r₈) (p₂ p₅ r₂ r₈) (p₂ p₇ r₁ r₅) (p₂ p₇ r₁ r₈)
 (p₂ p₇ r₃ r₅) (p₂ p₇ r₃ r₈) (p₂ p₈ r₂ r₅) (p₃ p₄ r₁ r₇) (p₃ p₅ r₂ r₄) (p₃ p₅ r₂ r₇)
 (p₃ p₅ r₄ r₇) (p₃ p₆ r₁ r₄) (p₃ p₇ r₁ r₄) (p₃ p₈ r₂ r₄) (p₃ p₈ r₂ r₇) (p₃ p₈ r₄ r₇)
 (p₃ p₉ r₁ r₇) (p₄ p₇ r₁ r₅) (p₄ p₇ r₁ r₈) (p₄ p₇ r₃ r₅) (p₄ p₇ r₃ r₈) (p₄ p₈ r₅ r₇)
 (p₄ p₉ r₁ r₇) (p₅ p₇ r₄ r₈) (p₅ p₈ r₂ r₄) (p₅ p₈ r₂ r₇) (p₅ p₈ r₄ r₇) (p₆ p₇ r₁ r₄)

The total number of valid solutions is 96.

If it is only necessary to find a valid solution, any element in V will suffice. However, if the optimal solution is desired, all the elements in V must be rated in some fashion, and then compared to all other elements in V in order to select the optimal solution.

2.2 Layout Abstraction

As the first step towards applying this topological algorithm to a real problem, bounds are introduced to the space. Next, a method is needed for describing the relative positions of both the end points of the nets and the arcs that join those points. Rather than imposing some sort of grid system to specify the locations of the points and arcs, the (bounded) space can be divided into regions based on the positions of the points with respect to each other and the boundaries. The layout space can then be represented as a planar adjacency graph, and the connecting arcs as paths through the regions between the points. Figure 2.4 shows the example problem of 2.1 with boundaries and regions imposed. Notice that as few regions as possible have been used to cover the entire bounded

space and yet still distinguish the individual arcs $r_1 - r_9$, $p_1 - p_9$. This is desirable because the process of finding a valid solution for a given set of nets, T , begins by generating the set, Q , of all possible paths for each two-point connection. These paths take the place of the arcs in the original formulation of this problem (arc sets $A_1 = \{r_1 \dots r_9\}$ and $A_2 = \{p_1 \dots p_9\}$ for this example). Depending on the number of regions covering the layout space, there may be more than one path for each of the original arcs. Fewer regions means that fewer paths are generated for each of the arcs. Also note that although the regions are all rectangular, they are not required to all be of a similar size.

Figure 2.4 nets N1 and N2 with boundaries and regions imposed

The adjacency graph which corresponds to the space in Figure 2.4 is shown in Figure 2.5. The vertices of the graph correspond to the regions in Figure 2.4. An edge exists between any two vertices which are physically adjacent in Figure 2.4. Each of the points $a_1 - a_3$, $b_1 - b_3$, is marked on one or more of these edges indicating that it lies on the boundary common to both of the regions (represented by the vertices) which the edge joins. Table 2.1 contains a set of paths which correspond to each of the arcs $r_1 - r_9$, $p_1 - p_9$ of the original formulation.

Figure 2.5 Adjacency graph for nets N_1 and N_2 Table 2.1 Paths for arc sets A_1 and A_2

arc	path	# alternate paths
r_1	(1)	7
r_2	(3 6 4)	3
r_3	(2 9 5)	0
r_4	(3 6)	8
r_5	(2 9 7)	5
r_6	(2 9 5 8)	7
r_7	(4 6)	8
r_8	(5 9 8)	5
r_9	(5 9 2 7)	7
p_1	(9)	7
p_2	(7 6 8)	3
p_3	(2 1 5)	0
p_4	(7 6)	8
p_5	(2 1 3)	5
p_6	(2 1 5 4)	7
p_7	(8 6)	8
p_8	(5 1 4)	5
p_9	(5 1 2 3)	7

The paths listed in column 2 of Table 2.1 are not the only paths which describe each arc. For example, arc r_1 can also be described by the path (3 4). These two paths (1) and (3 4) are completely equivalent because there are no obstacles on the boundaries between regions 1 and 3, regions 1 and 4, regions 2 and 3, or regions 5 and 4. The third column of Table 2.1 indicates the number of alternate paths for each of the arcs. The set, Q , of all possible paths for each two-point connection of each net in T , will most definitely contain such equivalent paths. However, it is only necessary to consider the shortest of any set of equivalent paths. Therefore, Q can immediately be reduced to the set Q' by detecting and removing all but the shortest of any set of equivalent paths that are generated.

In Figure 2.6, an obstacle, o_1 , has been placed at the corner of the regions 1, 3 and 4. In this situation, the paths (1) and (3 4) are not equivalent. Therefore, both paths must be part of Q' because there is no way to directly determine which path (if either) will be used in the final valid solution. Section 2.3 describes in more detail a general algorithm for detecting and removing equivalent paths.

Figure 2.6 introduction of the obstacle, o_1

For the original problem without obstacles (see Figure 2.4), Q is just the set of paths listed in Table 2.1, but the set Q (which includes equivalent paths) actually contains 100 more paths. Even for such a small set of two-point connections as in this example, a relatively large number of paths are generated for the initial set Q . The reduction heuristic of Section 2.6 combined with the solution selection heuristic of Section 2.5 will find a valid solution from the paths in Q as long as such a solution exists. However, most real problems to which this process would be applied are significantly larger. It is obvious that the time involved in generating Q and reducing it to a set of valid solutions very quickly becomes prohibitive as the number of nets increases. A method is needed to generate only a subset, Q_1 , of the paths in Q , which is of a workable size, yet still yields a valid solution. Such a method would preferably avoid including paths in the subset Q_1 which have little or no chance of being part of an optimal or near-optimal solution.

The optimality of a solution usually reflects short path length in some way. Since the probability is very low that a long, meandering path is part of an optimal or near-optimal solution, initial size of Q_1 can be significantly reduced if such paths are not generated in the first place. This can be accomplished by limiting the regions through which a path for a given two-point connection is permitted to travel. For example, if paths for the two-point connection $(a_1 a_3)$ are not permitted to travel through any regions below or to the right of regions 7 and 8 in Figure 2.4, then any path using regions 9 and/or 5 will not be generated. If similar restrictions are applied to the two-point connections $(a_2 a_3)$, $(b_1 b_3)$, and $(b_2 b_3)$, then Q_1 only contains paths for arcs r_1 , r_4 and r_7 of net N_1 , and arcs p_1 , p_4 and p_7 of net N_2 .

If equivalent paths are ignored, then Table 2.2 describes the case when Q_1 contains paths for arcs r_1 , r_4 , r_7 , p_1 , p_4 and p_7 .

Table 2.2 Paths for arcs in the reduced set Q_1

arc	path	end points	# crosses	crossing arcs
r_1	(1)	a_1, a_2	0	-
r_4	(3 6)	a_1, a_3	1	p_4
r_7	(4 6)	a_2, a_3	1	p_7
p_1	(9)	b_1, b_2	0	-
p_4	(7 6)	b_1, b_3	1	r_4
p_7	(8 6)	b_2, b_3	1	r_7

The algorithm described in Section 2.6 (which essentially discards the arc with the most crossing arcs until a solution is found) can be applied to the arcs in Table 2.2 to find a set of valid solutions as follows:

- arcs with maximum crossings:

$$\{r_4, r_7, p_4, p_7\}$$

- eliminate r_4
- arcs with no crossings:

$$\{r_1, p_1, p_4\},$$

- arcs with maximum crossings:

$$\{r_7, p_7\}$$

- cannot eliminate r_7 so eliminate p_7
- arcs with no crossings:

$$\{r_1, r_7, p_1, p_4\}$$

- the following valid solution exists:

$$(r_1, r_7, p_1, p_4)$$

An optimal valid solution (ie. $S_f = \{r_1, r_7, p_1, p_4\}$) was obtained even though a much smaller set of paths was initially considered. In this simple case, the reduction algorithm in itself was sufficient to find a good solution. However, this is not usually the case. There-

fore, the branch and bound algorithm for selecting a solution (see Section 2.5) would be used in conjunction with the reduction algorithm to find a solution.

2.3 Detection and Removal of Equivalent Paths

Two paths for a given two-point connection of a net are said to be equivalent if there is no obstacle inside the area bounded by the two paths. If K is a set of paths for the net $N = \{a_1 a_2\}$, then a path, k_1 , in K cannot be equivalent to any other path, k_2 , in K if there is an obstacle associated with any of the edges over which k_1 and k_2 travel, nor any of the edges enclosed by these two paths (see Figure 2.7).

a) equivalent paths

b) non-equivalent paths

Figure 2.7 Equivalent and non-equivalent paths

The adjacency graph of Figure 2.8 represents some arbitrary space in which the points of the net $N = \{a_1 a_2\}$ exist. If $k_1 = \{5 4 6 9 12\}$ and $k_2 = \{5 1 3 8 11 12\}$ are two paths for this net, the procedure for determining whether or not the k_1 and k_2 are equivalent is as follows:

1. Get a list of all edges in paths k_1 and k_2 .
2. Remove from this list the end edges with which the points a_1 and a_2 are associated.

3. If any of the edges in the list has an obstacle associated with it, stop because k_1 and k_2 are not equivalent. (For example, the edges between regions 4 and 6, regions 6 and 9, regions 9 and 12, regions 4 and 5, regions 5 and 1, regions 1 and 3, regions 3 and 8, regions 8 and 11, and regions 11 and 12 of Figure 2.9 must all be free of obstacles for k_1 and k_2 to be equivalent.)
4. Get a list of all edges enclosed by paths k_1 and k_2 .
5. If any of the edges in this new list has an obstacle associated with it, k_1 and k_2 are not equivalent. Otherwise, k_1 and k_2 are equivalent. (For example, the edge between regions 7 and 10 of Figure 2.9 must be free of obstacles for k_1 and k_2 to be equivalent.)

Figure 2.8 Example adjacency graph

net $N_1 = \{a_1 a_2\}$

path $k_1 = \{5 4 6 9 12\}$

path $k_2 = \{5 1 3 8 11 12\}$

Figure 2.9 Example paths k_1 and k_2

When two paths are found to be equivalent, the longer of the two paths is discarded since any procedure for finding the optimal solution would select the shorter of these two paths if given the option. For example, paths k_1 and k_2 of Figure 2.9 are identified as equivalent paths. Therefore, k_2 is discarded since it has one more vertex than k_1 .

2.4 Detection of Crossing Paths

The algorithm in [10] is a general algorithm for determining whether or not two paths on a graph cross. Briefly, the algorithm finds a set of vertices common to both paths, and assigns an order to the vertices in the two paths which are on either side of this common section of vertices. The two paths are said to cross if the following are true:

1. The start region of one path, k_1 , is earlier in the order than the start (or end) region of the other path, k_2 .
2. The end region of path k_1 is earlier in the order than the end (or start) region of path k_2 .

Figure 2.10 Crossing paths

For example, if k_1 and k_2 are the paths on the graph shown in Figure 2.10, the common section, r , consists of the regions 9 and 10. The regions on either side of r in both paths are then $s_1 = 8$ and $e_1 = 17$ for path k_1 , $s_2 = 5$ and $e_2 = 16$ for path k_2 . Starting at s_1 , the order

moving counter-clockwise around r is s_1, s_2, e_1, e_2 . Since s_1 is before s_2 , and e_1 is before e_2 , paths k_1 and k_2 cross. However, if k_1 and k_2 are the paths shown in Figure 2.11, the common section, r , consists of the regions 8 and 9. The regions on either side of r in both paths are therefore $s_1 = 7$ and $e_1 = 10$ for path k_1 , $s_2 = 16$ and $e_2 = 12$ for path k_2 . Starting at s_1 , the order moving counter-clockwise around r is s_1, e_1, s_2, e_2 . Since s_1 and e_1 are both before either s_2 or e_2 in this case, paths k_1 and k_2 do not cross.

Figure 2.11 Non-crossing paths

2.5 Optimal Solution Selection

In certain cases, it may be necessary to use an arc which is clearly sub-optimal to connect two points (ie. there exists at least one shorter arc connecting the same two points) to ensure that at least one feasible solution can be found. Q' often contains a great number of such sub-optimal arcs. When this is the case, the quality of a solution set of arcs, S_f , chosen randomly from Q' often may not be acceptable. Therefore, a more selective process for choosing a solution is required. Branch and bound techniques can be applied to select the solution set of arcs, S_f , from Q' . Classical branch and bound algorithms construct complete solutions from partial solutions, terminating when the optimal complete solution is found. The branching function operates on an overall-best basis,

guiding the movement through the partial solutions towards the next partial or complete solution which has the lowest cost of all nodes expanded to that point. The bounding function defines what constitutes the best complete solution that can be obtained from any partial solution.

Even though this process is more efficient than a straight enumeration of all possible solutions, it can still take a long time to find the optimal solution. This is especially true when Q is large, and the set of valid solutions (V) is small compared to the set of all possible solutions (P) for the problem. A greedy, depth-first approach to branching will generally yield a solution more quickly, especially if any valid solution will suffice. A more optimal solution can be obtained if the searching process (using the greedy, depth-first branching) is forced to continue until a complete solution which is within an acceptable tolerance range is found. However, this could also result in long processing times if the first sets of partial and complete solutions that are explored do not contain a near optimal solution.

An approach which combines the classic and depth-first approaches to branching should make it possible to find good solutions in a reasonable time frame. This combined approach employs the depth-first branching scheme to search the set of partial and complete solutions which is currently best. When the cost of the best solutions in this set reach a certain pre-determined range, a branch to the subset containing the overall best solution is executed. The new set is then searched, once again employing the depth-first branching scheme. The process continues until an optimal or near-optimal solution is found.

The branching function calculates the costs of partial and complete solutions based on the costs of their constituent arcs. A simple cost measure for an arc is its length. This cost could be based on other measures as well, such as the number of crossing arcs. The cost of a partial or complete solution is then just the sum of the costs of its constituent arcs plus the best possible cost associated with each net which is not yet included in the partial solution. A cost of infinity is associated with any partial or complete solution that is in-

valid (ie. two or more of the constituent arcs of the solution cross).

2.6 Reduction Heuristic

For small problems like the example of Figure 2.1, the sets of arcs which describe all possible two-point connections are small enough that it is possible to simply enumerate solutions until a valid solution is found. This method is feasible even if all valid solutions must be found in order to determine the optimal solution for the problem. However, for larger problems, there may simply be too many arcs for an exhaustive approach to find a solution in a practical amount of time.

Table 2.3 arc sets A_1 and A_2

arc	end points	# crosses	crossing arcs
r_1	a_1, a_2	2	p_5, p_8
r_2	a_1, a_2	4	p_4, p_5, p_7, p_8
r_3	a_1, a_2	5	p_3, p_5, p_6, p_8, p_9
r_4	a_1, a_3	3	p_2, p_4, p_9
r_5	a_1, a_3	5	p_1, p_3, p_5, p_6, p_9
r_6	a_1, a_3	7	$p_2, p_3, p_5, p_6, p_7, p_8, p_9$
r_7	a_2, a_3	3	p_2, p_6, p_7
r_8	a_2, a_3	5	p_1, p_3, p_6, p_8, p_9
r_9	a_2, a_3	7	$p_2, p_3, p_4, p_5, p_6, p_8, p_9$
p_1	b_1, b_2	2	r_5, r_8
p_2	b_1, b_2	4	r_4, r_5, r_7, r_8
p_3	b_1, b_2	5	r_3, r_5, r_6, r_8, r_9
p_4	b_1, b_3	3	r_2, r_4, r_9
p_5	b_1, b_3	5	r_1, r_3, r_5, r_6, r_9
p_6	b_1, b_3	7	$r_2, r_3, r_5, r_6, r_7, r_8, r_9$
p_7	b_2, b_3	3	r_2, r_6, r_7
p_8	b_2, b_3	5	r_1, r_3, r_6, r_8, r_9
p_9	b_2, b_3	7	$r_2, r_3, r_4, r_5, r_6, r_8, r_9$

Since the one factor which distinguishes the elements in V (the set of valid solutions) from P (the set of all solutions) is that no arc of a net crosses an arc which joins points of a different net, it seems reasonable that the number of times each arc crosses an arc of another net be used as a measure of the chances of that arc being present in the final solution. Table 2.3 contains all the arcs in Figures 2.2 and 2.3. The last two columns in the table contain the number of arcs of other nets which cross each arc, and a list of those arcs which cross each arc.

Table 2.3 seems to indicate that the number of valid solutions available is limited when one of the arcs used has a large number of crossing arcs. (eg. There are only six valid solutions which contain arc p_6 , but there are thirty-four which contain arc r_1 .) Therefore, it is more likely that a valid solution will be found if arcs with fewer crosses are considered. This indicates that the number of crossings arcs for each arc can be used to reduce the original arc sets (A_1 and A_2 in the example) for each net before seeking a solution.

If Q is defined to be the set of all arcs for all two-point connections for every net in T , then one method for reducing the number of arcs for each net is to eliminate arcs from Q until those that remain do not cross any arc associated with a different net. As long as there is at least one arc for each net remaining in Q , it will be possible to find at least one valid solution. If Z is defined to be the subset of Q whose arcs do not cross any arcs of other nets, then a straightforward approach to finding Z is the following:

Algorithm 1.0:

1. Get a list of all arcs which have the maximum number of crossing arcs.
2. If there is only one arc with the maximum number of crossings, remove that arc from consideration and reduce the crossing counts of all other arcs which it crossed, unless removing this arc will prevent any solution from being found. If there is more than one arc with the maximum number of crossings, remove the first one in the list and reduce the crossing counts of all other arcs which it crossed, again unless removing it will prevent any solution from being found. In the latter case, try the next arc in the list until either an arc is removed or no arc can be removed. If no arc can be removed, stop

because no solution is possible by removing the arc with maximum crossing arcs.

3. Get a list of all arcs which no longer cross any other arc (ie. find Z).
4. If Z contains enough arcs to completely connect all nets, stop because there is at least one combination of arcs in Z which is a valid solution. Otherwise, start again at step 1.

Applying algorithm: 1.0 to the example of Figure 2.1 with the arcs given in the order listed in Table 2.3, a subset, Z, of arcs which contains at least one valid solution is found as follows:

- arcs with maximum crossings:
 $\{r_6, r_9, p_6, p_9\}$
- eliminate r_6
- $Z = \{ \}$
- arcs with maximum crossings:
 $\{r_9\}$
- eliminate r_9
- $Z = \{ \}$
- arcs with maximum crossings:
 $\{r_3, r_5, r_8, p_6, p_9\}$
- eliminate r_3
- $Z = \{ \}$
- arcs with maximum crossings:
 $\{r_5, r_8\}$
- eliminate r_5
- $Z = \{ \}$
- arcs with maximum crossings:
 $\{r_8\}$
- eliminate r_8
- $Z = \{p_1, p_3\}$
- arcs with maximum crossings:

$\{r_2\}$

- eliminate r_2
- $Z = \{p_1 p_3\}$
- arcs with maximum crossings:
 $\{r_4 r_7\}$
- eliminate r_4
- $Z = \{p_1 p_3 p_4 p_9\}$
- arcs with maximum crossings:
 $\{r_7\}$
- cannot eliminate r_7 ; eliminate p_2, p_6 , and p_7
- $Z = \{r_7 p_1 p_3 p_4 p_9\}$
- arcs with maximum crossings:
 $\{r_1\}$
- cannot eliminate r_1 ; eliminate p_5 and p_8
- $Z = \{r_1 r_7 p_1 p_3 p_4 p_9\}$
- the following valid solutions exist:
 $(r_1 r_7 p_1 p_4), (r_1 r_7 p_3 p_4), (r_1 r_7 p_1 p_9), (r_1 r_7 p_3 p_9)$

In the case where the majority of arcs in Q have a large number of crossing arcs, the amount of time required to find a subset, Z , which contains a valid solution may be prohibitive. When this occurs, the reduction process can be stopped at a point when the reduced set of arcs, Q' , is a manageable size. A selection heuristic can then be applied to Q' to find a solution within a practical time frame. The number of crossing arcs for each arc in Q' may be used to help rate the arcs for the purposes of the selection heuristic.

One obvious shortfall of considering only the number of crossings is that in many cases, short path length is highly desirable. If only the number of crossings is considered, it is conceivable that a very desirable short path will be eliminated from future consideration if there are many paths which cross it. To avoid this problem, the cost associated

with a path should reflect its length as well as the number of crossing paths. This will cause paths which are both long and have a large number of crossing paths to be discarded first. Paths which are long but have few crossing paths, or short but have a large number of crossing paths would then have a better chance of being kept to participate the selection of a solution.

Chapter 3: Application of the Topological Single-Layer Routing Algorithm to Leaf Cell Routing

The detailed routing of a leaf cell circuit generally takes place after the optimum transistor placement for the cell has been determined and transistor sizing has been performed so that its main concern is to find the optimum set of paths to complete the connections of the circuit (see Figures 3.1 and 5.1). In this chapter, it is shown to be possible to reduce the leaf cell routing problem to a single-layer routing problem so that the topological single-layer routing (TSLR) algorithm can be applied. In this implementation, only leaf cells which use the line-of-diffusion (LOD) style of layout are considered.

Figure 3.1 Topological single-layer routing as part of the leaf cell synthesis problem

3.1 Reduction of the Leaf Cell Routing Problem to a Single Layer Routing Problem

In leaf cell routing, there are usually three routing layers in which connections are implemented - diffusion, polysilicon and metal1. In some advanced technologies, a local interconnect layer is also available. This application of the TSLR algorithm to leaf cell routing solves the problem with and without using the local interconnect layer.

In the LOD layout style, power (VDD) and ground (VSS) nets are carried by horizontal metal1 wires at the top and bottom of the bounded area which is the physical layout space for the cell. These power and ground wires extend the entire length of the layout space. The maximum vertical pitch of the cell is usually fixed, but the length of the cell is not usually given a fixed value, although there is a minimum length associated with the cell. (ie. The technology being used specifies a minimum spacing for neighboring transistor gates in a diffusion line, and a minimum spacing between transistors when there is a break in the diffusion line. Therefore, the minimum length of the cell is based on these two rules and on the number of transistors in the cell and the number of breaks in the lines of diffusion.) This minimum length is often sacrificed in an effort to keep the cell from exceeding the fixed maximum vertical pitch.

The layout space between the VDD and VSS wires can be perceived in terms of the following five horizontal areas: (Refer to Figure 3.2)

1. Top track area: the portion of the layout space below the metal1 VDD wire at the top of the layout space and above the p-type diffusion line
2. Pdiff area: the portion of the layout space occupied by the p-type diffusion line
3. Centre track area: the portion of the layout space between the p-type and n-type diffusion lines
4. Ndiff area: the portion of the layout space occupied by the n-type diffusion line
5. Bottom track area: the portion of the layout space below the n-type diffusion line and above the metal1 VSS wire at the bottom of the layout space

Figure 3.2 Five horizontal areas of the layout space

The first and last of these areas may or may not be present. If transistor sizing has resulted in a range of p (n) transistor widths, the top (bottom) track area exists, but there is only space available in this area for routing above (below) any p (n) transistor whose width is less than the width of the largest p (n) transistor.

Each of the five areas is further divided into a number of horizontal slices called tracks through which horizontal wires are routed. The number of tracks in the $pdiff$ and $ndiff$ areas depends on the transistor widths. There is a maximum number of tracks available for the other three areas combined (ie. maximum vertical pitch of the cell less the pitch of the $pdiff$ and $ndiff$ areas), but the actual number of tracks in each area is not fixed until the final set of paths for every connection is determined.

Of the four available routing layers, only the metal1 layer has complete freedom of horizontal and vertical movement through all five areas of the layout space. The diffusion layer is mainly used to implement the p -type and n -type transistors, but it is also used to make connections between neighboring transistors of the same type. Connections imple-

mented by long diffusion wires are undesirable because the diffusion layer is highly resistive. Connections between the diffusion and local interconnect layers are made without the use of a contact cut, but a contact cut is required to make a connection between the metal1 and local interconnect layers. Since polysilicon over diffusion forms a transistor, poly is only used to implement connections in the top, centre and bottom track areas. Local interconnect (LI) connects directly to both polysilicon and diffusion without using a contact, so although it can be used to implement connections in any of the five areas of layout space, it cannot be used to implement connections across either of the diffusion lines unless the sections of diffusion over which the LI wire passes are part of the connection.

The restrictions on the use of the diffusion, poly and LI layers, make it difficult to devise any leaf cell routing algorithm which uses all available routing layers to make connections. However, the leaf cell routing problem can be reduced to the problem of routing only those connections which are implemented in metal1 if as many connections as possible are partially or completely implemented in one or a combination of the restricted routing layers. It is to this metal1 routing problem that the TSLR algorithm can be applied. The following sections describe the rules which reduce the complete connection set to the set of only metal1 connections.

3.1.1 Diffusion Connections

In addition to the implicit connections between neighboring transistors of the same type which have a (drain-source) diffusion nets in common (see Figure 3.3), diffusion wires are used to implement some of the power and ground connections. Any p-transistor VDD diffusion net can be connected through a contact to the metal1 VDD wire at the top of the cell with a diffusion wire originating from the top-most p diffusion track. The diffusion contact is placed directly under the metal1 VDD wire. Similarly, any n-transistor VSS diffusion net can be connected through a contact to the metal1 VSS wire at the bottom of the cell with a diffusion wire originating from the bottom-most n diffusion track. Again,

the diffusion contact is placed directly under the metal1 (VSS) wire. Figure 3.4 contains examples of VDD and VSS connections implemented in diffusion.

Figure 3.3 Diffusion connection in neighboring transistors

Figure 3.4 Power and ground connections in diffusion; common gate connection in poly

3.1.2 Poly Connections

A transistor pair consists of a p transistor and an n transistor that are vertically aligned in the transistor placement order for the cell. Vertical poly wires are used to con-

nect common gate nets within transistor pairs. (see Figure 3.4)

Horizontal poly wires through the centre track area are used to connect common gate nets of neighboring transistor pairs if the following conditions are met:

1. There are no breaks in either the p or n diffusion lines between the neighboring transistor pairs with the common gate nets (ie. the p transistors of the two pairs share a common drain-source diffusion, and the n transistors of the two pairs also share a common diffusion). This is because the space associated with the break in the diffusion line is often used to effect a change in layers for one or both of the diffusion terminals on either side of the break (ostensibly for the purposes of creating I/O ports).
2. The net carried by the shared p diffusion is not the same net carried by the shared n diffusion.

The following configurations of the common gate net between neighboring transistor pairs are completed with horizontal poly wires:

1. gates of both the p and n transistors of both pairs are involved (see Figure 3.5 a)
2. gates of the p transistors of both pairs and the gate of the n transistor of one of the pairs are involved
OR
gates of the n transistors of both pairs and the gate of the p transistor of one of the pairs are involved (see Figure 3.5 b)
3. only the gates of the p transistors of both pairs are involved
OR
only the gates of the n transistors of both pairs are involved (see Figure 3.5 c)
4. gates of very wide p and n transistors which have been folded are involved (see Figure 3.5 d)

If there is a break in only one of the diffusion lines between neighboring transistor pairs with a gate net in common because the diffusion nets carried by the drain and source diffusions differ, horizontal poly wires can still be used to complete connections in any of the four configurations listed above. Such connections are permitted as long as neither of the nets carried by the transistor diffusions on either side of the break is the same as the net carried by the shared transistor diffusion of the other type. (see Figure 3.6)

Figure 3.5 Poly connections: a) neighboring common gates, b) neighboring common p or n gate, c) neighboring common p gates or n gates. d) folded transistor gate

Figure 3.6 Common gate connection across a diffusion break

If there is a break in only one of the diffusion lines between neighboring transistor pairs with a gate net in common due to a missing transistor in one of the pairs, horizontal poly wires can still be used to complete connections of the last three configurations listed above. However, this is only permitted when there is no break in the diffusion line for which transistors exist in both pairs, and when the diffusion net carried by the existing transistor of the pair which has a missing transistor differs from the net carried by the shared transistor diffusion of the other type. (see Figure 3.7)

Figure 3.7 Neighboring common gate connection when one pair is missing a transistor

3.1.3 LI Connections

Vertical LI wires are used to connect common diffusion nets (p source diffusion to n source diffusion or p drain diffusion to n drain diffusion) within transistor pairs. (see Figure 3.8)

Figure 3.8 Common p to n diffusion connection in LI

Horizontal wires through the centre, top or bottom track areas are used to make common drain-gate or gate-source connections between neighboring transistor pairs if the following conditions are met:

1. The nets carried by the source diffusions of the p and n transistors of the pair whose gate is involved in a drain-gate connection are different.
2. The nets carried by the drain diffusions of the p and n transistors of the pair whose gate is involved in a gate-source connection are different.

The following configurations of the common drain-gate net between neighboring transistor pairs are completed with horizontal LI wires:

1. the drains of both the p and n transistors of one pair, and the gates of both the p and n transistors of the other pair are involved (see Figure 3.9 a)
2. the drains of both the p and n transistors of one pair and only the gate of the p transistor of the other pair are involved

OR

the drains of both the p and n transistors of one pair and only the gate of the n transistor of the other pair are involved (see Figure 3.9 b)

3. only the drain of the p transistor of one pair and the gate of the p transistor of the other pair are involved

OR

only the drain of the n transistor of one pair and the gate of the n transistor of the other pair are involved (see Figure 3.9 c)

The following configurations of the common gate-source net between neighboring transistor pairs are completed with horizontal LI wires:

1. the gates of both the p and n transistors of one pair, and the sources of both the p and n transistors of the other pair are involved (see Figure 3.10 a)
2. the gates of both the p and n transistors of one pair and only the source of the p transistor of the other pair are involved

OR

the gates of both the p and n transistors of one pair and only the source of the n transistor of the other pair are involved (see Figure 3.10 b)

3. only the gate of the p transistor of one pair and the source of the p transistor of the other pair are involved

OR

only the gate of the n transistor of one pair and the source of the n transistor of the other pair are involved (see Figure 3.10 c).

A combination of horizontal and vertical LI wires through the centre track area is used to connect common drain-source diffusion nets in neighboring transistors of opposite type (see Figure 3.11).²

2. Recall that common drain-source diffusion nets in neighboring transistors of the same type are implicitly connected through shared diffusion areas.

Figure 3.9 LI connections: a) common drain to common gate, b) common drain to gate, c) drain to gate

Figure 3.10 LI connections: a) common gate to common source, b) common gate to source, c) gate to source

Figure 3.11 LI connection of neighboring diffusions of opposite type

A combination of horizontal and vertical LI wires through the top/bottom track area is used to connect one of the sets of common drain and source nets of very wide p/n transistors which have been folded (see Figure 3.12)..

Figure 3.12 LI connection of equipotential diffusion terminals in a folded transistor

3.2 Overview of the Algorithm

The topological single-layer routing algorithm presented in Chapter 2 operates on an adjacency graph which models the layout space. Therefore, the first step in applying the algorithm to the metal routing problem is the division of the layout space into rectangular regions to accurately represent the contents of the cell. Ultimately, the solution set of metal paths obtained from this algorithm will be converted to a symbolic layout representation that will be compacted to produce the final layout of the leaf cell. It is therefore essential that the division of the space into regions be fine enough to make the conversion

relatively straight-forward. Unfortunately, this is at odds with the need to use as few regions as possible so that the set of all paths for all nets does not exceed a manageable size. In addition, it is difficult to use a graph representation which places the end points of a net between more than one pair of vertices. This is unavoidable if the end points (pins) are placed at the corner of three or more regions. If the pins are placed either directly on a region, or on the boundary between two regions, there is no difficulty in determining which vertex or pair of vertices are to be used when only the pins of a net are known.

Because of these problems, the implementation of the TSLR algorithm is in two passes. In the first pass, the adjacency graph models the layout space in just enough detail to distinguish paths for all the metal connections. Pins are placed on vertical boundaries to divide the space around a pin into left and right sides. All multi-pin nets are divided into two-point connections, and a minimum set of these connections which completely describes all the metal nets is generated. The paths found for each of these two-point connections are restricted to a practical subset of all possible paths for that connection. This is to prevent the total number of paths in the set of all paths for all metal nets, Q , from becoming too large to handle. Few, if any, equivalent paths are generated, but any that are generated are subsequently removed. The reduction heuristic presented in Section 2.6 of Chapter 2 is used to reduce Q to a suitable size for the modified branch and bound algorithm (see Section 2.5 of Chapter 2) to select a set of paths for the solution, S . In the second pass, a more detailed adjacency graph is constructed based on both the original adjacency graph, and the pins used by the first-graph paths in S . In this graph, pins are located on horizontal boundaries to divide the space around a pin into top and bottom sides. Therefore, after the second pass, it will be known how all paths approach (from the top, bottom, left or right sides) or pass a given pin. This knowledge greatly simplifies the mapping of the solution set of paths to a symbolic layout representation.

A set of paths on the second graph are generated for each of the paths in S . Although these second-graph paths contain more vertices, they represent passage through the

same physical sections of the layout space as the first-graph paths for which they were generated. Again, equivalent paths are detected and removed from the set of all second-graph paths, Q_f . The modified branch and bound algorithm is then applied to select a set of second-graph paths for the final solution, S_f . If Q_f is initially too large for branch and bound algorithm, the reduction heuristic is applied to reduce Q_f before a solution is chosen.

Although the conversion of S_f to a symbolic layout representation is not actually part of the TSLR algorithm, the paths in S_f are not completely specified until the following information is known for each path:

1. how the path passes through each of its constituent regions (ie. horizontally, vertically, or some combination of both)
2. in what order (horizontal and vertical) the path passes through a region with respect to other paths from other nets which pass through the same region.

The practice of placing pins on vertical boundaries in the first pass and horizontal boundaries in the second pass of the TSLR algorithm makes it relatively easy to determine this information as part of the conversion (to a symbolic representation) process for a specific compactor.

The remainder of this chapter contains a detailed description of all aspects of the two-pass implementation of the TSLR algorithm for solving the problem of routing metal connections in leaf cells (see Figure 3.13). To assist with the understanding of the procedures described in this chapter, the next chapter takes an example leaf cell through the entire set of procedures.

Figure 3.13 Process flow of the topological single-layer leaf cell routing algorithm

3.3 Adjacency Graph Structure for the First Pass

As part of the process of reducing the leaf cell routing problem to the metal routing problem, the layout space is coarsely divided into five horizontal areas. To get the rectangular regions of layout space which will translate into the vertices of the adjacency graph, the layout space is further divided into a series of vertical slices. A vertical slice corresponds either to a source (drain) terminal of a transistor pair, or to the section of space between transistor pairs when there is a break in one or both of the diffusion lines. The total number of vertical slices for the leaf cell is then $2t + s + 2$, where t is the number of transistor pairs in the circuit, and s is the number of breaks between transistor pairs. The extra two slices correspond to the columns of space before the first transistor pair and after the last transistor pair. Figure 3.14 shows an example leaf cell layout with the horizontal areas and vertical slices superimposed.

Figure 3.14 Horizontal and vertical slicing imposed on the layout space

Except for the vertical poly wires for the gate nets of the transistor pairs, there is no obvious visible indication of the vertical slicing on the layout space. The slice information

is kept in the form of an ordered list of interval boundaries (from left to right). Each vertical slice has a left and right interval boundary associated with it.

There are six types of regions which are used to cover the entire layout space in the first pass. These regions represent space in one or more of the five horizontal areas, and in one or more of the vertical slices. They are the following:

1. Top track region - routing space in the top track area
2. Pdiff region - routing space predominantly in the pdiff area, but also in the top track area
3. Centre track region - routing space predominantly in the centre track area, but also in the other four areas
4. Ndiff region - routing space predominantly in the ndiff area, but also in the bottom track area
5. Bottom track region - routing space in the bottom track area
6. Spacer region - routing space in any or all five areas

Figure 3.15 Initial first graph region set for a transistor pair

The transistor placement of the leaf cell circuit can be expressed in terms of two ordered sets, S_p and S_n , of p-type and n-type transistors, respectively, which describe the transistor placement from left to right. A transistor pair consists of a p transistor from S_p and an n transistor from the corresponding position in S_n . Initially, each transistor pair has a set, R_i , of regions associated with it (see Figure 3.15). R_i consists of twelve regions of the following types:

- two spacer regions: one between the current and previous transistor pairs (region 1 in Figure 3.15), and one between the current and next transistor pairs (region 12 in Figure 3.15)
- two top track regions: one above the source side of the p transistor (region 2 in Figure 3.15), and one above the drain side of the p transistor (region 7 in Figure 3.15)
- two pdiff regions: one for the source side of the p transistor (region 3 in Figure 3.15), and one for the drain side of the p transistor (region 8 in Figure 3.15)
- two centre track regions: one below the source side of the p transistor (region 4 in Figure 3.15), and one below the drain side of the p transistor (region 9 in Figure 3.15)
- two ndiff regions: one for the source side of the n transistor (region 5 in Figure 3.15), and one for the drain side of the n transistor (region 10 in Figure 3.15)
- two bottom track regions: one below the source side of the n transistor (region 6 in Figure 3.15), and one below the drain side of the n transistor (region 11 in Figure 3.15)

Each of these regions has a left and right interval boundary associated with it, so that there is an initial set of five interval bounds associated with each transistor pair.

Recall that the arcs connecting the pins of a net are represented as paths on the adjacency graph. These paths are simply ordered lists of vertices (regions) which start and end at the regions associated with two pins of the net. There are three types of pins (named for the layer on which they originate) for the end points of a metal1 net:

1. Poly pin - a junction between the poly and metal1 layers
 - it may be associated with top, centre, or bottom track regions
2. Diffusion pin - a junction between a (p or n) diffusion layer and the metal1 layer
 - it may be associated with pdiff or ndiff regions

3. Locint pin
 - a junction between the local interconnect and metal1 layers
 - it may be associated with top, centre, or bottom track regions

Poly pins are used for the gate terminals of a transistor pair. They are depicted along the centre interval boundary of the transistor pair, and are therefore always associated with two regions. Diffusion pins are used for the source and drain terminals of a transistor pair. They are depicted directly on the diffusion region associated with the appropriate terminal of the transistor pair, and are therefore associated with all interval boundaries associated with that diffusion region. Locint pins are also used for the source and drain terminals of a transistor pair. LI is used to extend the source and drain terminals into the top, centre and bottom track areas of the layout space. They are depicted along either of the interval boundaries immediately to the left or right of the centre interval boundary of the transistor pair, and are therefore always associated with two regions.

Each transistor pair has an initial set of pins, P_i , associated with it (see Figure 3.15). P_i consists of sixteen pins of the following types:

- eight locint pins: two for the source net of the p transistor (pins a and c in Figure 3.15), two for the drain net of the p transistor (pins k and m in Figure 3.15), two for the source net of the n transistor (pins d and f in Figure 3.15), and two for the drain net of the n transistor (pins n and p in Figure 3.15)
- four diffusion pins: one for the source net of the p transistor (pin b in Figure 3.15), one for the drain net of the p transistor (pin l in Figure 3.15), one for the source net of the n transistor (pin e in Figure 3.15), and one for the drain net of the n transistor (pin o in Figure 3.15)
- four poly pins: two for the gate net of the p transistor (pins g and h in Figure 3.15), and two for the gate net of the n transistor (pins i and j in Figure 3.15)

There are a number of physical situations associated with a given p/n transistor pair which cause variations in either or both of the initial region and pin sets, R_i and P_i respectively. The variations manifest themselves in three ways: region elimination, pin elimination, and region merging.

3.3.1 Region Elimination

Region elimination occurs when the top and/or bottom horizontal areas do not exist. In such a case, the top and bottom track regions (regions 2 and 7 and/or regions 6 and 11 of Figure 3.15) are removed from the set R_i for each transistor pair. The present software implementation of the TSLR algorithm does have the ability to eliminate top and/or bottom track regions on a per transistor pair basis during the construction of the adjacency graph. However, the remainder of the algorithm is currently implemented based on the assumption that the transistors are all of equal size. The full effects of transistor sizing will be accounted for in a future version of the software.

3.3.2 Pin Elimination

Pin elimination occurs in three situations. When LI is not used for routing the leaf cell circuit, the eight locint pins (pins a, c, d, f, k, m, n, and p in Figure 3.15) are removed from the set P_i for each transistor pair. When the net with which a pin is associated does not need to be routed in metal1, that pin is removed from the set P_i . (eg. pins a, b and c in Figure 3.15 are removed from P_i if the source net of the p transistor does not need to be routed in metal1) When the net carried by two centre pins of the same type is associated with the same terminal (S, G or D) in both the p and n transistors of a transistor pair, a single common centre pin for the net is sufficient. Therefore, one of the two centre pins is removed from the set P_i for that transistor pair. (eg. pin i in Figure 3.15 is removed from P_i if both transistors have the same metal1 gate net).

3.3.3 Region Merging

Region merging occurs when the boundary between two of the regions in R_i is not necessary. This happens in the following different situations:

1. When the nets for the same terminal (source, gate or drain) of the p and n transistors of a transistor pair do not need to be routed in metal1, the centre regions on either side of the pins in P_i which are associated with these nets are merged, and the pins are removed

from P_i . For example, regions 1 and 4 in Figure 3.15 are merged into the single region 1 in Figure 3.16 when neither of the source terminals of p and n transistors need to be routed in metal1.

Figure 3.16 Region merging due to pin elimination

2. When the source (drain) net of a p-type or n-type transistor in one transistor pair is the same as the drain (source) signal in the same-type transistor of the previous (next) pair, there is no break in the line of diffusion between the two transistor pairs. In such a case, the drain and source regions of the transistor which carry the common net are merged, and the spacer region between the two transistor pairs is altered. For example, regions 8 and 14 in Figure 3.17 are merged into the single region 8 in Figure 3.18 when the drain net of the p transistor of the first pair is the same as the source net of the p transistor of the second pair.
3. When there are no breaks in either of the p or n diffusion lines between two transistor pairs, the drain and source regions carrying the common nets are merged for both types of transistors, and the spacer region is eliminated entirely. For example, regions 8 and 14 in Figure 3.17 are merged into the single region 8 in Figure 3.19, and the regions 10 and 16 in Figure 3.17 are merged into the single region 10 in Figure 3.19. Notice that region 12 in Figure 3.17 does not exist in Figure 3.19

②	⑦	⑫	⑬	⑱
③	⑧		⑭	⑲
④	⑨		⑮	⑳
⑤	⑩		⑯	㉑
⑥	⑪		⑰	㉒

Figure 3.17 Example region map

②	⑦	⑬	⑱
③	⑧		⑲
④	⑨	⑫	⑮
⑤	⑩		⑯
⑥	⑪		⑰

Figure 3.18 Region merging due to shared diffusion

Figure 3.19 Region merging due shared diffusions of both types

Figure 3.20 Region merging due to a missing n transistor

4. When one of the transistors of a transistor pair is missing, the centre track regions for the source and drain sides of that pair are merged with the pdiff (ndiff) and top (bottom) track regions of the missing p (n) transistor. For example, when the n transistor for the first transistor pair in Figure 3.17 is missing, regions 4, 5 and 6 in Figure 3.17 are merged into the single region 4 in Figure 3.20, and regions 9, 10 and 11 in Figure 3.17 are merged into the single region 7 in Figure 3.20. Figure 3.21 depicts the case where the p transistor for the first transistor pair is missing. Note that as a result of region merging, it is possible that a region be associated with more than two interval bounds.

Figure 3.21 Region merging due to a missing p transistor

3.3.4 Adjacency Graph

The adjacency graph for the first pass consists of one vertex for each region in the final versions of R_i (ie. the version which takes into account all the variations just described) for each transistor pair. Only the pins in the final versions of P_i (ie. the version after all pins have been eliminated) for each transistor pair are associated with vertices of the adjacency graph.

3.4 Determining the Minimum Set of Two Point Connections for the Metall Nets

A net in a leaf cell circuit is essentially a set of transistor terminals which must be connected together. Each of these terminals has at least one pin associated with it.

3.4.1 Sub-nets

A two-point metall connection (sub-net) is made when there are pins of the net associated with two or more transistor pairs. However, not necessarily all pins of a net form sub-nets with all other pins of that net for the following reasons:

1. More than one pin may be available for a single transistor terminal. Therefore, it is not necessary to form sub-nets with both pins. For example, pins a, b, and c in Figure 3.15 are all pins associated with the source terminal of the p transistor. (They are connected via LI wires connected to the source diffusion of the transistor.) Any other pin carrying the same net as these three pins need only form a sub-net with one of them.
2. For each net which has terminals in more than two transistor pairs, it is not always necessary to form sub-nets between pins on transistor pairs which are far apart. For example, let p_1 be a pin for net S_1 associated with transistor pair 3 in a given leaf cell circuit, and let p_2 and p_3 be S_1 pins associated with transistor pairs 6 and 13 respectively in the same circuit. If a sub-net can be implemented between pins p_1 and p_2 , and another sub-net can be implemented between pins p_2 and p_3 , then there is no point in forming a sub-net between pins p_1 and p_3 because the probability of using the p_1 - p_3 sub-net instead of the p_2 - p_3 sub-net to completely connect all three pins is very low.
3. Certain terminals of a given net may already be connected on one of the other available routing layers. Even if pins exist for these terminals (so that connections to other pins of the net which have not been previously connected on another layer may be formed), there is no need to duplicate the pre-routed connections in metall.

3.4.2 OR-Groups

Pins for the metall net of the circuit are divided into sets called or-groups. An or-group is a set of pins belonging to the same net which are already connected on a routing layer other than metall. Sub-nets are formed between pins in consecutive or-groups. The

purpose of grouping the pins in this manner is to provide a number of options for connecting two terminals of a net. In the final solution, only one sub-net out of all the possible combinations of sub-nets formed from the pins in a given pair of or-groups is required; all other sub-nets between the pins of these two or-groups are redundant.

Pins in an or-group are associated with a single terminal of a transistor pair, and have at least one interval boundary in common. The or-groups for each net are ordered from left to right according to the numbers of the interval boundaries. There are two types of or-groups: gate or-groups and source/drain or-groups. These are discussed in the following sub-sections.

3.4.2.1 Gate OR-group

This type of or-group contains pins associated with the gate terminal of a transistor pair. If the n and p transistors of the pair carry the same net on their gate terminals, then there is a maximum of three pins in the or-group. These include a pin in the centre track area, one pin in the top track area if the top track area exists, and one pin in the bottom track area if the bottom track area exists. If the n and p transistors of the pair carry different nets on their gate terminals, two separate or-groups are formed, each containing a maximum of two pins. An or-group for the gate net of a p transistor has a pin in the centre track area, and one in the top track area if the top track area exists. An or-group for the gate net of an n transistor has a pin in the centre track area, and one in the bottom track area if the bottom track area exists.

3.4.2.2 Source/Drain OR-group

This type of or-group contains pins associated with the source (drain) terminal of a transistor pair. If the n and p transistors of the pair carry the same net on their source (drain) terminals, then there is a maximum of five pins in the or-group. These include a pin in each of the pdiff, centre track and ndiff areas, one pin in the top track area if the top track area exists, and one pin in the bottom track area if the bottom track area exists. If the

n and p transistors of the pair carry different nets on their source (drain) terminals, two separate or-groups are formed, each containing a maximum of three pins. An or-group for the source (drain) net of a p transistor has a pin in each of the pdiff and centre track areas, and one pin in the top track area if the top track area exists. An or-group for the source (drain) net of an n transistor has a pin in each of the ndiff and centre track areas, and one pin in the bottom track area if the bottom track area exists.

3.4.3 Minimum Set of Sub-nets for a Net

The minimum set of sub-nets for a net is formed between the or-groups for the net according to the following rules:

1. Pins in an or-group are only permitted to form sub-nets with pins in the next or-group for the net unless the or-group under consideration contains only a single pin. In such a case, the single pin is permitted to form sub-nets with the next two or-groups if they exist, and if the second or-group is near enough to the single-pin or-group. In the case where the last or-group for the net contains only a single pin, that or-group is permitted to form sub-nets with the previous two or-groups as long as the leftmost of these two groups is within three transistor pairs of the last or-group.³
2. One or more or-groups of virtual pins are introduced to the circuit for a net when consecutive or-groups are more than three transistor pairs apart. These virtual-pin or-groups are placed between the original two widely-spaced or-groups in the ordered list of or-groups for the net. Then, for the purposes of forming sub-nets, these or-groups are treated just like normal or-groups. (See Section 3.4.4 for more details.)
3. Pins in an or-group which lie in the top track area may not form sub-nets with pins in another or-group which lie in either the ndiff or bottom track areas.
4. Pins in an or-group which lie in the pdiff area may not form sub-nets with pins in another or-group which lie in the bottom track area.
5. Pins in an or-group which lie in the centre track area may form sub-nets with any pin in another or-group.

3. Observations from tests of various leaf cell circuits run on the prototype software have determined that pins separated by more than three transistor pairs cause an unacceptably large number of paths to be generated.

6. Pins in an or-group which lie in the ndiff area may not form sub-nets with pins in another or-group which lie in the top track area.
7. Pins in an or-group which lie in the bottom track area may not form sub-nets with pins in another or-group which lie in either the top track or pdiff areas.
8. Sub-nets which are already implemented on another routing layer are not included in the set of sub-nets for the net.

The algorithm which generates all the paths for each sub-net requires that one pin of each sub-net be designated as the start pin, and the other as the end pin (see the Section 3.5.1). The choice of which pin of a sub-net is to be designated as the start pin may be important. The following rules are used to assign a start pin for each sub-net:

1. If only one pin of a sub-net is a poly pin, that pin must be designated as the start pin.
2. If both pins of a sub-net are poly pins, either of the pins may be designated as the start pin.
3. If neither pin of a sub-net is a poly pin, either of the pins may be designated as the start pin.

Poly pins are preferred as start pins to allow as many sub-nets as possible to take advantage of the path generation rule which terminates paths at diffusion regions (see Section 3.5.4).

3.4.4 Using Virtual Pins to Split Long Sub-nets

Even though the number of regions covering the layout space has been kept to a minimum, if the pins of a sub-net are associated with transistor pairs that are more than three pairs apart, the number of paths generated for that sub-net is too large to handle in a practical amount of time. Splitting the long sub-net into one or more smaller sub-nets can solve this problem. These sections of the long nets will be recombined (ie. eliminating the virtual pins) after the reduction algorithm of Section 3.7 has reduced the number of paths to be considered during solution selection. To accomplish the splitting, one or more sets of virtual pins are introduced at pre-defined intervals between the pins of the original long sub-net.

A virtual pin is simply a point in the layout space where two sections of metal wire can meet. A set of virtual pins is defined in a single vertical slice and consists of one pin for each horizontal area that exists (ie. there are five pins per set if both top and bottom track areas exist). Every path for the long sub-net normally passes through the vertical slice containing the virtual pins. Therefore, having a pin in every available horizontal area ensures that no path for the complete long sub-net, regardless of which horizontal area of the vertical slice it passes through, is excluded.

The present version of the software is capable of using virtual pins to split these long sub-nets. However, the locations of the virtual pins are currently chosen manually. A future version of the software should include a procedure for determining the locations of virtual pin sets

3.5 Finding All Paths For A Sub-net

Figure 3.22 Example adjacency graph

The paths for a sub-net are generated by expanding the adjacency graph into a tree using the regions associated with the start and end pins of the sub-net as the root and leaf nodes, respectively. To keep the size of the set, Q , of all paths of all nets to a reasonable size, paths which have a very low probability of being used in the solution set, S , are simply not generated. This is accomplished by restricting the expansion process in a number of ways. The following sub-sections describe the techniques used to control the expansion process in order to limit the number of paths generated for each metal sub-net. To assist with the following explanations, a generic net, $g = (s\text{-pin } e\text{-pin})$, is defined to represent any and all sub-nets of a given leaf cell circuit. In addition, the adjacency graph of a fictitious circuit is depicted in Figure 3.22 to provide a source for any examples that are necessary.

3.5.1 Restricting the Boundaries of the Search Space for the Paths of a Sub-net

Since many of the sub-nets in a given leaf cell circuit are localized in one area of the adjacency graph, it is not always necessary to consider every vertex (region) in the graph when finding paths for individual sub-nets. This is especially true when the pins of the sub-net are physically close to each other on the graph. For each sub-net, a bounding box of regions which enclose the pins of the sub-net is created. Vertices outside this bounding box are temporarily disconnected from the graph so that when the graph-to-tree expansion takes place, the paths generated for this sub-net will not contain any regions outside this bounding box.

For each sub-net, the leftmost and uppermost pins (not necessarily different pins) are determined. If both pins are in the same horizontal area, the start pin is designated as the uppermost. Normally the pins of a sub-net are not in the same vertical slice. However, this does happen when the diffusion terminals (source or drain) of the p and n transistors in a transistor pair need to be connected, and the LI layer is not used to make connections. In such a case, the designation of one the pins as leftmost is irrelevant. We use the relative positions of the pins of the sub-net to construct a bounding box for the sub-net according

to the definitions in the following sub-sections.⁴

3.5.1.1 Top Row

The top row of the bounding box is defined to be the row of regions that is one horizontal area above the row of regions which contains the region associated with the uppermost pin. If the uppermost pin has two regions associated with it, the top row of the bounding box is the row of regions in the horizontal area directly above the row of regions containing the uppermost of the two regions associated with that pin. For example, let s-pin = 10, and e-pin = 22 of Figure 3.22. The top row of the bounding box is then {2 1 6 13 18 25 27}.

In the case where the (uppermost) region associated with the uppermost pin lies in the topmost horizontal area of the layout space, the top row of the bounding box is defined to be the row which contains the (uppermost) pin region.

3.5.1.2 Bottom Row

The bottom row of the bounding box is defined to be the row of regions that is immediately below the region associated with the pin which is not uppermost. If this pin has two regions associated with it, the bottom row of the bounding box is then the row of regions in the horizontal area directly below the row of regions containing the bottom-most of the two regions associated with that pin. For example, let s-pin = 6, and e-pin = 13 of Figure 3.22. The bottom row of the bounding box is then {2 4 9 16 21 23 29}.

In the case where the (bottom-most) region associated with the pin which is not uppermost lies in the bottom-most horizontal area of the layout space, the bottom row of the bounding box is defined to be the row which contains the (bottom-most) pin region.

4. The present version of the software is capable of widening the bounding box by one region in all directions to cause more paths to be generated for each sub-net. The purpose for this is to provide a larger initial set of paths, Q , from which to choose a solution when no solution is found with the current size of bounding box. However, the present version of the software does not make use of this feature; it simply terminates when no solution is found with the smallest bounding boxes being used for all sub-nets.

3.5.1.3 Left Column

If the pins of the sub-net are not located in the same vertical slice, then if the start pin of the sub-net is leftmost, the left column of the bounding box is defined to be the column of regions which contains the leftmost region associated with the start pin. If the end pin of the sub-net is leftmost, then the left column of the bounding box is the column of regions immediately to the left of the column of regions which contains the leftmost region associated with the end pin. For example, let s-pin = 14, and e-pin = 7 of Figure 3.22. The left column of the bounding box is then {5 1 7 3 4}.

In the case where the leftmost region lies in the leftmost column of the graph, the left column of the bounding box is defined to be the leftmost column of the graph.

3.5.1.4 Right Column

If the pins of the sub-net are not located in the same vertical slice, then if the start pin of the sub-net is not the leftmost of the two pins, the right column of the bounding box is defined to be the column of regions which contains the rightmost region associated with the start pin. If the end pin of the sub-net is not leftmost, then the right column of the bounding box is the column of regions immediately to the right of the column of regions which contains the rightmost region associated with the end pin. For example, let s-pin = 1, and e-pin = 9 of Figure 3.22. The right column of the bounding box is then {17 18 19 20 21}.

In the case where the rightmost region lies in the rightmost column of the graph, the right column of the bounding box is defined to be the rightmost column of the graph.

3.5.1.5 Left and Right Columns For Pins in the Same Vertical Slice

If the pins of the sub-net are located in the same vertical slice, the left column of the bounding box is defined to be the column of regions which is three columns to the left of the column of regions containing both pins. In the case where the leftmost column of the graph is less than three columns to the left of the column of regions containing both pins,

the left column of the bounding box is defined to be the leftmost column of the graph.

For pins in the same vertical slice, the right column of the bounding box is defined to be the column of regions which is three columns to the right of the column of regions containing both pins. In the case where the rightmost column of the graph is less than three columns to the right of the column of regions containing both pins, the right column of the bounding box is defined to be the rightmost column of the graph.

3.5.2 Preventing the Generation of Tightly Zigzagging Paths

A tightly zigzagging path is one which moves one direction along a row (column) of regions, moves over to the neighboring row (column) and proceeds in the opposite direction, then moves over to the next neighboring row (column) and proceeds in the original direction again, and so forth. For example, the path consisting of the regions { 10 12 17 18 13 6 11 14 19 22 } of Figure 3.22 is a tightly zigzagging path. No self-respecting routing algorithm would consider such paths when choosing the final solution, so their generation is prevented.

As the adjacency graph is expanded into a tree, a list, consisting of all the neighboring regions of all regions currently in the path being expanded which are not part of the path themselves, is kept. The decision concerning which branch to expand to find the next region in the path is made by examining the neighboring regions, N , of the last region in the path (to this point). Any region in N which is not also part of the path to this point, and not a neighbor of any previous region in the partial path is a valid choice for the next region in the path. For example, let $s\text{-pin} = 16$, and $e\text{-pin} = 5$ of Figure 3.22. If p is the partial path { 22 18 13 14 }, then the neighboring regions, N , are { 25 17 19 12 6 }. The neighboring regions of the last region of the partial path are { 11 13 15 19 }, but since region 13 is already part of p , and region 19 is part of N , the next region may only be 11 or 15. If none of the neighboring regions of the last region of a partial path is a valid choice for the next region, the partial path is abandoned since it is not a valid route for the sub-net.

3.5.3 Early Detection and Termination of Invalid Paths

In some cases it is possible to detect when the path currently being expanded will turn out to be invalid. This is useful because it means a partial path which will be invalid can be abandoned without having to expand it to the end. It is possible to detect whether a partial path still has any possibility of becoming a valid path by checking whether any of the regions associated with the end pin of the sub-net remain available.

An end pin region becomes unavailable as soon as it is included in the set of neighboring regions of the partial path, N . Once no end regions remain available for a partial path, there is no valid way of completing that path for the sub-net, so the partial path can be immediately abandoned. For example, let $s\text{-pin} = 1$, $e\text{-pin} = 10$ of Figure 3.22. The end pin regions are $\{14\ 19\}$. If p is the partial path $\{7\ 8\ 15\ 20\}$, then p will turn into an invalid path because both end pin regions are neighbors of one or more of the regions already in the partial path - region 14 is a neighbor of region 15, and region 19 is a neighbor of region 20.

3.5.4 Termination of Paths at Diffusion Regions

If the end terminal associated with a set of sub-nets is the source or drain terminal of a transistor, then at least one of the pins in the set of pins, P_d , for that terminal will be associated directly with the diffusion region for that transistor. When the adjacency graph is expanded for a sub-net that has as an end pin one of the other (non-diffusion) pins in P_d , the region associated with the diffusion pin in P_d is included as one of the valid regions for the leaves of the tree. (Normally, the leaf regions for the tree for a given sub-net include only the regions associated with the end pin of the sub-net.)

The diffusion region is included in the list of end regions for the sub-net because the diffusion region is always a neighbor of the regions associated with the end pin for the sub-net. Therefore, if a path passes through the diffusion region on its way to one of the regions associated with the end pin, then by definition, there must be sufficient space for a metal wire to be routed through that diffusion region, and if so, there is room for a direct

metal-to-diffusion connection. For example, let e-pin = 18 of Figure 3.22, then the end pin regions are 24 and 27. Pin 19 (on region 25) is associated with the drain terminal of the same p transistor. Therefore, region 25, which is a neighbor of both regions 24 and 27, is included as one of the end regions for all sub-nets that end with e-pin = 18.

The ability to cut short paths at the diffusion region becomes much more valuable if the end pin for a sub-net is associated with a source (drain) terminal that is common to both transistors in a pair when LI is used to make connections. In such a case, an LI wire already connects the source (drain) terminals of the p and n transistors together so it is possible to terminate a path for a sub-net at the diffusion regions associated with the terminal of either transistor. For example, let s-pin = 10, and e-pin = 22 of Figure 3.22. If p is the partial path { 14 15 20 }, then one possible choice for the next region on the path is 28 which has pin 21 associated with it. Since pins 21 and 22 both belong to the set of pins for the drain terminal of the n transistor in the last pair, a valid way to complete p is to connect to pin 21 on region 28, since 21 and 22 are connected with LI.

One consequence of this process of prematurely terminating paths at diffusion regions is that the set of all paths for a net may contain duplicates. This happens when the same path is generated for two different sub-nets whose end pins belong to the set of pins associated with the same terminal. There is no easy way to detect when a path is not unique as it is being generated. However, the uniqueness of a path can be detected when the path is added to the list of paths for the net. If the path to be added already belongs to the list of paths for the net, it is simply discarded.

A double hashing storage scheme for the paths is employed to speed up the process of determining if a path is unique (and must, therefore, be kept). The index into the first hash table is based on the number of the end region of the path. Each location in the first hash table indexes a location in the second hash table based on path length. Each location in the second hash table then indexes a location in a large path storage table. At each location in this path storage table, there is a list of paths which end with the same region, and

are all of the same length. A new path is only added to the list in the proper location of the path storage table if it does not match any of the paths already in that list.

3.5.5 Storing All the Unique Paths of All Nets

The system used to store the paths for the sub-nets as they are generated is designed to efficiently determine the uniqueness of a path. However, this system is awkward to use everywhere else. A more natural method of storing the paths for each sub-net is to index each path according to the pins associated with its start and end regions. Each unique path of each sub-net is therefore stored using a double hash scheme similar to the one described in Section 3.5.4. However, this time the hashing is based on the numbers of the start and end pins. For example, to retrieve a list of all the paths for the net g from this storage system, s -pin is used to get the list of all the different sub-nets which start with s -pin. Then, e -pin is used to retrieve the subset of these sub-nets which also end with e -pin. This hashed set of unique paths for each sub-net of each net is simply the set Q .

3.6 Detection and Removal of Equivalent Paths

The detection and removal of the equivalent paths of a net is carried out as described in Section 2.3 of Chapter 2. However, the process of checking the regions enclosed by the two paths for obstacles (ie. pins of another net) requires further elaboration. If Q' is defined to be the subset of Q which contains no equivalent paths, then the process of reducing Q to Q' by examining subsequent pairs of regions in two paths, k_1 and k_2 , for equivalency is as follows:

1. If the regions being compared are the last regions of their respective paths, then k_1 and k_2 are equivalent.
2. If the two regions being compared are the same, skip to the next region in each path.
3. If the two regions being compared are neither the same nor are they in the same column (see Figure 3.23), select the path which contains the leftmost of the two regions and skip regions in that path until a region is found that is in the same column as the region of the other path

4. If the regions being compared are not the same, but are in the same column (see Figure 3.24), check all regions in the column between k_1 and k_2 for pins. If there are pins associated with any of these regions, k_1 and k_2 are not equivalent, and no further comparison is necessary. Otherwise, skip to the next region in each path.

When any two paths are found to be equivalent, only the shorter of the two is included in Q' .

Figure 3.23 Checking for pins between regions in different columns

Figure 3.24 Checking for pins between regions in the same column

3.7 Reduction of the Path Set Prior to Solution Selection

Even though paths with a low probability of being in the solution set, S , are not generated, and equivalent paths are removed, the total number of paths in Q' may still be too large to ensure that a feasible solution is found in a reasonable period of time. When this is the case, some of the paths in Q' which do not have a high probability of being present in an optimal solution can be discarded prior to the application of the branch and bound solution algorithm of Section 3.8.

3.7.1 Assigning a Cost to Each First-Graph Path

The paths which belong to the solution set, S , must not cross any path for a sub-net associated with a different net since such a cross manifests itself as short of the two nets in the physical layout. In addition, a short path connecting two end points is usually preferable to a longer path connecting the same two points. Therefore, the probability of forming an optimal solution containing a given path in Q' should be based on both the path's length and the number of paths which cross it. This probability can be reflected in a cost which is the product of these two items. Paths which have high costs (ie. a lower probability) associated with them are therefore the first to be discarded when the reduction heuristic of Chapter 2 is applied.

3.7.2 Recording Crossing Paths

Since the procedure for counting the crossings for a path is potentially very time-consuming, it is essential that it be as efficient as possible. There are a number of ways to minimize the number of paths to which any one path, p_0 , in Q' is compared, and thus speed up this process.

Since paths associated with the same net cannot cause a short, it is not necessary to check for crossings amongst paths of the same net. Therefore, the paths in Q' are grouped by net. Each path in a net group is only compared to paths which are in a different net group. In addition, crossing path totals for both paths being compared are updated when a

cross is detected. Therefore, it is only necessary to check for a crossing between any pair of paths once. For example, if $T = \{N_1 N_2 N_3 N_4\}$ is a set of nets, then to completely check for all path crossings, first compare paths associated with net N_1 to paths associated with net N_2 , then net N_3 , and finally net N_4 . Then, it is only necessary to compare paths associated with net N_2 to paths associated with net N_3 and N_4 , and finally, to compare paths associated with net N_3 to paths associated with net N_4 . This process essentially reduces the number of comparisons by half.

There are pairs of paths which cannot possibly cross because one path is entirely to the left or right of the other (ie. does not overlap the other) on the adjacency graph (see Figure 3.25). Detecting the fact that a pair of paths cannot possibly cross obviates the need for a more detailed check for a crossing. A group tree data structure [11] has been implemented for each net group which contains all the paths of all other net groups. The interval $(x_1 x_2)$ for a path is based on the interval boundaries for the regions at the left and right extremes of the path. x_1 is defined to be the leftmost interval boundary of the leftmost region in the path. x_2 is defined to be the rightmost interval boundary of the rightmost region in the path. These structures are used to efficiently find the set of all paths which overlap a given path p_0 . It is only necessary to perform a detailed check for a crossing between paths which overlap.

Figure 3.25 Non-overlapping paths

If all the regions in a path, p_0 , are located in the uppermost (bottom-most) row of regions in the adjacency graph, then if there are at least two tracks available for routing through each of the regions in that row,⁵ p_0 cannot possibly cross any other path. If there is only one track available for routing through each of the regions in the row, then it is only necessary to check for common regions between p_0 and any other path to determine whether or not the paths cross.

Finally, if two paths have no regions in common they cannot possibly cross. Therefore, the detailed check for a crossing is only carried out when the two paths have at least one region in common.

The algorithm described in Section 2.4 of Chapter 2 is employed to perform the detailed check for a crossing between two paths on the adjacency graph. Refer to Appendix B for examples of the ways a cross can occur when a single region is common to both paths, and when there are at least two consecutive regions common to both paths.

In the special case where the common section, r , includes the start or end region of one or both paths, it is not possible to directly apply this algorithm. Even though it may appear on the graph that two paths in this situation do not cross, they may physically still do so. For example, the common section between paths k_1 and k_2 of Figure 3.26 b) is the set of regions {15 16 17}. Since k_1 connects the pins b and d, and k_2 connects the pins a and c, it is obvious from the map of the layout space in Figure 3.26 a) that the two paths do cross. Since pin c is associated with two regions, (the other is region 18), if path k_2 is temporarily extended to include this other region (18), the algorithm in Section 2.4 may still be applied to determine if k_1 and k_2 cross. Therefore, whenever the common section between two paths includes an extreme region, r_x , of one or both of the paths, if the pin associated with r_x has a second region associated with it, that second region is added to the path(s) before any checks for a crossing are performed. If there is no second pin associated with r_x , the paths do not cross.

5. this includes any spacer regions which occur when there is a break in the p (n) diffusion line between transistor pairs

Figure 3.26 Using extended paths to check for a crossing

Whenever a crossing is detected between two paths, the number of crossing paths is increased by 1 for both paths. In addition, the paths are placed in each other's list of crossing paths so that the reduction heuristic can directly reduce the crossing counts of only those paths which cross the paths being eliminated.

3.7.3 Reducing the Set of First-Graph Paths

Once the crossing counts of all paths have been recorded, the average number of paths per sub-net, A_s , is calculated. The reduction of the path sets of each sub-net is then carried out as follows:

1. calculate the following parameters for the sub-net:

- a) total number of paths for the sub-net
 - b) maximum length path
 - c) minimum length path
 - d) maximum number of crossings for a path
 - e) minimum number of crossings for a path
2. calculate the length which represents the twentieth percentile
 3. calculate the number of crossings which represents the twentieth percentile
 4. sort the paths in order of decreasing cost
 5. discard paths for the sub-net until either there are no more than A_s paths remaining, or until all the paths which remain fall below the twentieth percentiles of length and/or crossings

3.7.4 Eliminating Virtual Pins to Recombine Long Sub-nets

Upon completion of the reduction of Q' , the remaining paths associated with sub-nets where at least one end point is a virtual pin must be recombined to form the original long paths from which these sub-nets were derived. This is done for two reasons. First, it would be necessary to significantly complicate the branch and bound solution algorithm (see Section 3.8) to recognize redundant paths when virtual pins are involved. Second, the order of the branch and bound searching is related exponentially to the number of sub-nets, but linearly to the number of paths. That is, the order of the searching is proportional to some function of the total number of paths to be considered raised to the power of another function of the total number of sub-nets to which the paths belong. Therefore, it is more desirable to have a large number of paths than a large number of sub-nets. For example, 3000^{12} is much larger than 3000^6 , so it is more likely that a solution would be found quickly when there are only six sub-nets even though those sub-nets have three thousand paths associated with them.

There are essentially four ways in which two sections of a long net are recombined upon removal of a virtual pin.

directly from region 10 of path k_1 to region 12 of path k_2 to create the long path {2 5 8 10 12 13 17}, completely discarding the regions 7 and 11 which create the unwanted loop. A very good compactor may be able to detect and remove such an open loop, but it is more efficient to eliminate it at this point

3.7.4.4 Merge With Closed Loop Removal

This is used when the two sections create an unwanted closed loop if combined using the direct or modified direct merge. For example, since the path k_1 of Figure 3.27 d) which connects pin 0 to virtual pin 14 and the path k_2 of Figure 3.27 d) which connects pin 9 to virtual pin 14 both end at region 9, a modified direct merge could be used to combine the two paths. However, this would result in a long path which contains a forbidden closed loop near the location of the eliminated virtual pin. Such a loop can cause problems for a compactor, therefore, a better merge is effected by moving directly from region 7 of path k_1 to region 11 of path k_2 to create the long path {0 4 7 11 14}, completely discarding the regions 8, 9 and 10 which create the closed loop.

3.8 Selecting the Solution Set of First-Graph Paths

3.8.1 Branch and Bound Selection

The modified branch and bound algorithm which is used to find the solution sets of paths uses a set of four parameters to control how the search operation proceeds and when it terminates. They are the following:

1. GREEDY - when this parameter is true, the search operation terminates as soon as the first complete solution is found
2. EPSILON - this parameter (≥ 1) is used to calculate the maximum acceptable cost of a complete solution. When the search operation is not greedy, it will terminate only when a complete solution that has a cost in the range

$$\text{optimum cost} \leq x \leq \text{EPSILON} * \text{optimum cost}$$
is found

- 3. DELTA - this parameter together with INCR control the branching operation (see relation 3.8.i)
- 4. INCR - this parameter together with DELTA control the branching operation (see relation 3.8.1)

(3.8.1) dive when $C \leq N * DELTA + INCR$, otherwise branch

where C = least expensive child node being considered

N = least expensive node in the tree

The DELTA and INCR parameters work in concert to determine when to dive and when to branch. Essentially relation 3.8.1 means that if the cost of the node reached by diving is not too high relative to the cost of the least expensive node that can be reached by branching, continue to dive. DELTA and INCR determine the value of "not too high".

When the greedy parameter is not true, the EPSILON parameter controls when the search for a solution terminates. EPSILON is essentially a quality control parameter. A small EPSILON guarantees that the search will not terminate until a (near optimal) complete solution is found. Larger values of epsilon often mean less time is taken to find a solution, but the quality of that solution is usually sub-optimal.

In the prototype software, the cost of an individual path for a given sub-net is simply its length. Therefore, the cost of a complete solution is simply the total of the lengths of all the paths in that solution. The cost of a partial solution is a total of the lengths of the known costs, and an estimate of the minimum possible cost of the remaining paths required for a complete solution. This minimum cost estimate is obtained by totalling the lengths of the shortest path (or paths) which completely connect all the terminals of each net in the circuit. For example, if p_1 through p_6 are paths for the net N_1 in Figure 3.28, and if the cost of $p_1 \leq$ the cost of $p_2 \leq$ the cost of p_3 etc., then the best possible solution (ie. the minimum cost solution) for connecting all terminals of net N_1 is the total cost of the paths p_1 , p_2 and p_3 . It is not possible to connect the terminals of N_1 for a lower cost.

net $N_1 = \{a_1 a_2 a_3 a_4\}$

Figure 3.28 Example net N_1

The feasibility of a (partial or complete) solution is determined when the children of a node are generated. If no path from any of the sub-nets in a given or-group can be added to the current partial solution without causing that partial solution to become infeasible, then no children are generated for the node corresponding to the current partial solution. Feasibility of a solution is based on the available capacity for routing wires, and on whether or not the paths in partial solution cross. A solution is infeasible if the number of paths required to pass through a region exceeds the number of available routing tracks. Only paths which contribute distinct horizontal segments through a region are counted. For example, paths p_1 and p_2 in Figure 3.29 count as only one track because both paths can use the same horizontal track. A solution is infeasible if any two paths in the solution which carry a different net cross. For example, path p_3 in Figure 3.30 cannot be part of a feasible solution which contains paths p_1 and p_2 in Figure 3.30, because path p_3 crosses path p_1 .

Figure 3.29 Paths occupying the same horizontal segment

Figure 3.30 Paths which cross

3.8.2 Determining Available Tracks for the Solution

Once the solution set of first-graph paths is known, it is possible to determine how many horizontal tracks will be available for routing metal wires in each of the five horizontal track areas. The current version of the prototype software assumes that all transistors of the same type are the same size. Furthermore, transistor sizing has taken place so the number of available tracks in both the pdiff and ndiff horizontal areas is known. Therefore, it is only necessary to determine how many tracks will be required through the top, bottom and centre track areas.

Since the final solution is to be translated into a symbolic representation which will then be compacted, it is only crucial that the number of tracks be at least sufficient to route all the metall connections. Therefore, a simple method for determining the number of available tracks is to total the number of solution paths which travel through each of the top, bottom and centre track regions⁶, and then determine the number of tracks required for each area from the maximums of these totals. For example, if the maximum number of solution paths which travel through a centre track region is three, then three tracks must be available in the centre track area. The total number of available tracks for the solution is then the sum of the available tracks for each of the five horizontal track areas.

3.9 Adjacency Graph Structure for the Second Pass

The adjacency graph for the second pass represents a much finer subdivision of the layout space. One major difference between the first and second graphs is that a column of regions is introduced to represent the gate of a transistor pair in the second graph. In the first graph, the gate is simply a boundary between the source and drain columns of regions for the transistor pair. The regions in the second graph differ from the regions in the first graph in essentially two ways:

1. No region represents routing space in more than one horizontal area.
2. More than one region may represent routing space in the same horizontal area for a single vertical slice.

Finally, all pins in the second graph are associated with horizontal boundaries between regions. This is in contrast with the situation found in the first graph where the pins are either associated directly with a region, or with the vertical boundary between a pair of regions.

6. Spacer regions are ignored in this tabulation since all paths which pass through a spacer region must also travel through at least one other region which is not a spacer, and therefore confined to a single track area. Totals for centre track regions which span more than one track area are determined based on the previous and next regions in the path.

Each of the regions in the initial set of regions, R_i , associated with each transistor pair in the first graph is represented by one or more regions on the second graph so the initial set of regions, R_{fi} , associated with each transistor pair in the second graph (see Figure 3.31) consists of forty-one regions of the following types:

- two top track spacer regions: one between the current and previous transistor pairs (region 1 in Figure 3.31), and one between the current and next transistor pairs (region 37 in Figure 3.31)
- two pdiff spacer regions: one between the current and previous transistor pairs (region 2 in Figure 3.31), and one between the current and next transistor pairs (region 38 in Figure 3.31)
- two centre spacer regions: one between the current and previous transistor pairs (region 3 in Figure 3.31), and one between the current and next transistor pairs (region 39 in Figure 3.31)
- two ndiff spacer regions: one between the current and previous transistor pairs (region 4 in Figure 3.31), and one between the current and next transistor pairs (region 40 in Figure 3.31)
- two bottom track spacer regions: one between the current and previous transistor pairs (region 5 in Figure 3.31), and one between the current and next transistor pairs (region 41 in Figure 3.31)

These ten spacer regions represent the two spacer regions of the first graph (regions 1 and 12 of Figure 3.15)

- six top track regions: two above the source side of the p transistor (regions 6 and 7 in Figure 3.31), two above the gate of the p transistor (regions 17 and 18 in Figure 3.31), and two above the drain side of the p transistor (regions 26 and 27 in Figure 3.31)

These six top track regions represent the same section of top track area covered by the two top track regions of the first graph (regions 2 and 7 in Figure 3.15).

- five pdiff regions: two for the source side of the p transistor (regions 8 and 9 in Figure 3.31), one for the gate of the p transistor (region 19 in Figure 3.31), and two for the drain side of the p transistor (regions 28 and 29 in Figure 3.31)

These five pdiff regions represent the same section of pdiff area covered by the two pdiff regions of the first graph (regions 3 and 8 in Figure 3.15).

Figure 3.31 Initial second graph region set for a transistor pair

- nine centre track regions: three below the source side of the p transistor (regions 10, 11 and 12 in Figure 3.31), three below the gate of the p transistor (regions 20, 21 and 22 in Figure 3.31), and three below the drain side of the p transistor (regions 30, 31 and 32 in Figure 3.31)

These nine centre track regions represent the same section of centre track area covered by the two centre track regions of the first graph (regions 4 and 9 in Figure 3.15).

- five ndiff regions: two for the source side of the n transistor (regions 13 and 14 in Figure 3.31), one for the gate of the n transistor (region 23 in Figure 3.31), and two for the drain side of the n transistor (regions 33 and 34 in Figure 3.31)

These five ndiff regions represent the same section of ndiff area covered by the two ndiff regions of the first graph (regions 5 and 10 in Figure 3.15).

- six bottom track regions: two below the source side of the n transistor (regions 15 and 16 in Figure 3.31), two below the gate of the n transistor (regions 24 and 25 in Figure 3.31), and two below the drain side of the n transistor (regions 35 and 36 in Figure 3.31)

These six bottom track regions represent the same section of bottom track area covered by the two bottom track regions of the first graph (regions 6 and 11 in Figure 3.15)

Each of these regions has a left and right interval boundary associated with it, so that there is an initial set of six interval bounds associated with each transistor pair. The one extra interval boundary in the second graph (there are only five associated with the same group of regions in the first graph) is a result of representing the gate of the transistor pair as a column of regions instead of as a boundary between the columns of regions for the source and drain of the transistor pair.

The same initial set of sixteen pins, P_i , associated with the transistor pair in the first graph (see Figure 3.15) is associated with each transistor pair in the second graph (see Figure 3.31).

3.9.1 Region Elimination

The same physical situations which caused variations in either or both of the initial region and pin sets (R_i and P_i respectively) associated with a given p/n transistor pair in the first graph, cause corresponding variations in the initial region set (R_{fi}) for that transistor pair in the second graph. Top and bottom track regions (regions 1, 6, 7, 17, 18, 26, 27, 37 and regions 5, 15, 16, 24, 25, 35, 36, 41 in Figure 3.31) are eliminated if the top and bottom horizontal arcs do not exist. Whenever a pin is removed from P_i in the first graph, the two regions associated with that pin in the second graph are merged (eg. pins a, b and c in Figure 3.15 are removed from P_i if the p transistor's source net does not need to be routed in metall, so regions 6 and 7 in Figure 3.31 are merged, regions 8 and 9 are merged, and regions 10 and 11 are merged). Additional region merging occurs when the boundary between two regions in R_{fi} is not necessary for one of the following reasons:

②	⑪	⑫	⑲	⑳
③	⑫		㉑	㉒
④	⑬	⑬	㉒	㉓
	⑭		㉓	
⑤	⑮	⑭	㉔	㉕
⑥	⑯		㉕	㉖
⑦	⑰		㉖	㉗
⑧	⑱	⑮	㉗	㉘
	㉑		㉘	
⑨	㉒	⑯	㉙	㉚
⑩	㉓		㉚	㉛

Figure 3.32 Example second pass region map

1. When the source (drain) net of a p-type or n-type transistor in one transistor pair is the same as the drain (source) net in the same-type transistor of the previous (next) pair, there is no break in the line of diffusion between the two transistor pairs. In such a case, the drain and source regions of the transistor which carry the common net are merged, and the diffusion spacer region between the two transistor pairs is eliminated. If top track regions above the drain and source regions exist, they are also merged and the top track spacer between the two pairs is eliminated. Note that unlike in the first graph, the top track regions are separated by a horizontal boundary to indicate that there is a top track pin available for routing the metal net. For example, when the drain net of the p transistor of the first pair is the same as the source net of the p transistor of the second pair,

- spacer regions 22 and 23 of Figure 3.32 are eliminated
- regions 11 and 27 of Figure 3.32 are merged into the single region 11 of Figure 3.33
- regions 12 and 28 of Figure 3.32 are merged into the single region 12 of Figure 3.33
- regions 13 and 29 of Figure 3.32 are merged into the single region 13 of Figure 3.33
- regions 14 and 30 of Figure 3.32 are merged into the single region 14 of Figure 3.33

②	⑪			③⑧
③	⑫			③⑨
④	⑬			④⑩
	⑭			
⑤	⑮	②④	③①	④①
⑥	⑮		③②	④②
⑦	⑮		③③	④③
⑧	⑮	②⑤	③④	④④
	⑮		③⑤	
⑨	⑮	②⑥	③⑥	④⑤
⑩	⑮		③⑦	④⑥

Figure 3.33 Region merging due to shared diffusion

2. When there are no breaks in either of the p or n diffusion lines between two transistor pairs, the drain and source regions carrying the common nets are merged for both types of transistors, and the spacer regions are eliminated entirely. For example, when the drain nets of both transistors of the first pair are the same as the source nets of both transistors of the second pair,)

- spacer regions 22, 23, 24, 25 and 26 of Figure 3.32 are eliminated

- regions 11 and 27 of Figure 3.32 are merged into the single region 11 of Figure 3.34
- regions 12 and 28 of Figure 3.32 are merged into the single region 12 of Figure 3.34
- regions 13 and 29 of Figure 3.32 are merged into the single region 13 of Figure 3.34
- regions 14 and 30 of Figure 3.32 are merged into the single region 14 of Figure 3.34
- regions 15 and 31 of Figure 3.32 are merged into the single region 15 of Figure 3.34
- regions 16 and 32 of Figure 3.32 are merged into the single region 16 of Figure 3.34
- regions 17 and 33 of Figure 3.32 are merged into the single region 17 of Figure 3.34
- regions 18 and 34 of Figure 3.32 are merged into the single region 18 of Figure 3.34
- regions 19 and 35 of Figure 3.32 are merged into the single region 19 of Figure 3.34
- regions 20 and 36 of Figure 3.32 are merged into the single region 20 of Figure 3.34
- regions 21 and 37 of Figure 3.32 are merged into the single region 21 of Figure 3.34

②	⑪	⑳
③	⑫	㉑
④	⑬	㉒
	⑭	
⑤	⑮	㉓
⑥	⑯	㉔
⑦	⑰	㉕
⑧	⑱	㉖
	㉒	
⑨	㉓	㉗
⑩	㉔	㉘

Figure 3.34 Complete elimination of the spacer regions

21 of/de 2

PM-1 3 1/2" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

POWERS IN METHYLENE BLUE TESTING SINCE 1974

Micred

15001 CRAFTY ROAD & BURNINGWELL RD. BURNINGWELL, VA 22026 USA
TEL 812 486 7887 FAX 812 486 7887 TLA 8100000000

3. When one of the transistors of a transistor pair is missing, the centre track regions for the source and drain sides of that pair are merged with the pdiff (ndiff) and top (bottom) track regions of the missing p (n) transistor. For example, when the n transistor for the first transistor pair in Figure 3.32 is missing,

- regions 6, 7, 8, 9 and 10 in Figure 3.32 are merged into the single region 14 in Figure 3.35
- regions 16, 17, 18, 19, 20 and 21 in Figure 3.32 are merged into the single region 20 in Figure 3.35

Figure 3.36 depicts the case where the p transistor for the first transistor pair is missing. Note that as a result of region merging, it is possible for a region to be associated with more than two interval bounds.

Figure 3.35 Region merging due to a missing n transistor

Any differences between the set of pins associated with the first graph, P_i , and the final set of pins associated with the second graph, P_f , are due to the elimination of virtual pins, and the elimination of any pins in P_i which are not used by the paths in the solution,

S. When a pin is not used by any of the paths in the final solution, S_f , it is removed and the regions in the second graph associated with that pin are merged.

Figure 3.36 Region merging due to a missing p transistor

3.9.2 Adjacency Graph

The adjacency graph for the second pass consists of one vertex for each region in the final sets of R_{fi} (ie. the region sets which take into account all the variations just described) for each transistor pair. Only the pins in the final sets of P_{fi} (ie. the pin sets after all pins have been eliminated or added) for each transistor pair are associated with vertices of the adjacency graph.

3.10 Finding All Paths on the Second Graph for Each of the First-Graph Solution Paths

Paths generated on the second graph are simply more detailed versions of the solution paths on the first graph. The same process of expanding the graph into a tree is used to

generate these paths. However, the bounding which limits the number of regions that are considered for any one path is different. Each solution path is translated into a corresponding set of regions (vertices), R , in the second graph. Any vertex not contained in R is temporarily disconnected from the graph so that when graph-to-tree expansion of the second graph takes place, all paths generated will contain only regions which belong to R . In addition, the restriction on tightly zigzagging paths is lifted. The bounding is tight enough to prevent any truly meandering paths from being generated.

The translation of a solution path, p , to a corresponding set of second-graph regions, R , is accomplished by considering the regions in p one at a time, starting from one end of the path. First the regions on the second graph associated with the solution path region being considered are added to R . Then, the direction of the next region in p with respect to the solution path region currently being considered is determined. If the next region is above or below the current solution path region, the translation of the current region is complete. If, however, the next region is to the left or right of the current solution path region, any gate region on the second graph which lies between the second-graph regions associated with the current and next solution path regions must be added to R .

An example to help explain this transformation is depicted in Figure 3.37. The example assumes all pins (a through p) in the first graph of Figure 3.37 a) are used in the solution set of paths, S . Table 3.1 shows the set of regions in the second graph of Figure 3.37 b) which represent the regions in the first graph.

For example, if p is the set of regions {0 2 5 6 9 11 12} on the first graph in Figure 3.37 a), then the set of regions, R , which correspond to p on the second graph in Figure 3.37 b) is {3 4 5 6 10 11 15 16 18 23 24 26 27 31 32 33 34}. Notice that the gate regions 10, 11, 26, and 27 are included in R even though they do not have corresponding regions in the first graph. Regions 10 and 11 of the second graph are included because they are in between regions on the second graph (5, 6, and 15, 16 in Figure 3.37 b) which represent the regions 2 and 5 of the first graph. Regions 26 and 27 of the second graph are included

because they are in between regions on the second graph (23, 24 and 31, 32 in Figure 3.37

b) which represent the regions 9 and 11 of the first graph.

Figure 3.37 Example first graph to second graph translation

Table 3.1 Region correspondence between first and second graphs

first graph region	second graph regions
0	3, 4
1	0, 1, 2
2	5, 6
3	7, 8
4	13, 14
5	15, 16
6	17, 18
7	19, 20
8	21, 22
9	23, 24
10	29, 30
11	31, 32
12	33, 34
13	31, 32
14	39, 40
15	41, 42
16	43, 44
17	35, 36, 47

Once the set of regions, R , for a solution path, p , has been determined, the process of expanding the second graph into a tree to generate paths is identical to the expansion process used with the first graph with one minor exception. Gate regions can only be entered from the right or the left. Therefore, once one of a pair of gate regions associated with a pin has been chosen to be part of a path on the second graph, the other region associated with that pin must not be allowed to be part of that path. It is therefore placed directly into the list of regions which are neighbors of the regions already in the path.

Q_i is defined to be the set of all paths on the second graph for all of the solution paths in S .

3.11 Detection and Removal of Equivalent Paths

Two paths on the second graph are said to be equivalent if all of the following con-

ditions are met:

1. both paths were generated for the same solution path
2. both paths begin with the same region on the second graph
3. both paths end with the same region on the second graph
4. there is no pin inside the area between the two paths on the second graph

The procedure which checks for pins associated with region enclosed by two paths on the second graph is identical to the procedure described in Section 3.6 of this chapter. The subset of Q_f which contains no equivalent paths is defined to be Q_f' .

3.12 Applying the Reduction Heuristic to the Set of Second Graph Paths

In some cases, it may not be necessary to reduce Q_f' further before applying the modified branch and bound algorithm of Section 3.8 to select a final solution. However, when it is necessary to eliminate some of the second-graph paths, the reduction heuristic of section 3.7 is used to reduce Q_f' to a manageable size before the final solution is selected.

3.13 Selecting a Final Solution Set of Second-Graph Paths

The same branch and bound selection algorithm used to find the solution set of first-graph paths, S , may be used to find the final solution set of second-graph paths, S_f . However, since the paths on the second graph are simply more detailed versions of the paths of the acceptable solution S , a more greedy branching approach is often sufficient to find a good final solution. The advantage to using the greedy branching scheme is that a solution is usually found very quickly.

3.14 Mapping the Final Solution Into Symbolic Layout

Two final pieces of information are required before a final solution path is completely specified. They are the following:

1. the position of the path in the horizontal and vertical orders of its constituent regions with respect to all other solution paths which pass through those regions
2. how the path passes through each of its constituent regions (ie. horizontally, vertically, or a combination)

This information is most easily specified during the final conversion of the solution set of paths into a symbolic layout representation.

As part of the mapping process, each of the paths in the final solution set, S_t , is divided into horizontal and vertical segments, so that an order can be assigned to the horizontal (vertical) segments of each path which pass through a given region. In the current software implementation, horizontal and vertical path segments are sorted in ascending order. The procedure for determining the lesser⁷ of two paths, k_1 and k_2 , in either the horizontal or vertical order of a region, r , is a variation of the algorithm in [10] for detecting when two paths cross. The procedure determining the lesser path is as follows:

1. Find the extreme regions, s_x and e_x , of the common section of regions for k_1 and k_2 which includes r .
2. Find the regions s_1 and e_1 on either side of the s_x and e_x in path k_1 , and the regions s_2 and e_2 on either side of s_x and e_x in path k_2 .
3. Determine the lesser path according to the following rules:
 - If both s_1 and s_2 have values, then k_1 is less than k_2 if and only if s_1 is less than s_2 .
 - If s_1 (s_2) does not have a value, then if s_2 (s_1) is less than s_x , k_2 (k_1) is less than k_1 (k_2). Otherwise, use e_1 , e_2 , and e_x to determine which path is the lesser.
 - If both e_1 and e_2 have values, then k_1 is less than k_2 if and only if e_1 is less than e_2 .
 - If e_1 (e_2) does not have a value, then if e_2 (e_1) is less than e_x , k_2 (k_1) is less than k_1 (k_2).

Two paths can occupy the same horizontal track, t , (vertical routing column, c) in a region if either:

7. For horizontal segments, a path is the lesser of two when it is bottom-most. For vertical segments, a path is the lesser of two when it is leftmost.

1. both paths are associated with the pin assigned to track t (routing column c)
2. one path is entirely to the left of (below) the other path, and there is sufficient room for both.

A Cartesian grid is imposed on the layout space as part of the symbolic mapping process. Once the ordering of the path segments is complete, the pins (which are the end points of the solution paths) are assigned to specific tracks and routing columns in the symbolic representation. Tracks are associated with values along the y axis of the imposed grid, and routing columns are associated with values along the x axis of the imposed grid. The coordinates of each metal pin are set as follows:

1. Let p be the pin whose coordinates are to be set. Then, r is a region on the second graph associated with p , and m is the path which has r for either its first or last region.
2. If m does not contribute a vertical segment to r , determine on which side, s , of r (left or right) is the next region to r in m , and assign the x coordinate of p to be the number of the vertical routing column in r which is closest to s . For example, if $m = \{6\ 11\ 15\ 21\}$ is the path connecting pins a and c in Figure 3.38, and r is region 6, then s is the right side because region 11 is to the right of region 6. Therefore, the x coordinate of pin a is set to routing column 2.
3. If m contributes a vertical segment to r , find its position, o , in the ordered list of vertical path segments for r , and assign the x coordinate of p to be the number of the vertical routing column in r which corresponds to o . For example, if $m = \{16\ 17\ 22\ 27\ \dots\}$ is the path beginning with pin d in Figure 3.38, r is region 16, and o is the second position, then the x coordinate for pin d is set to routing column 5.
4. If m does not contribute a horizontal segment to r , determine on which side, s , (top or bottom) is the other second graph region associated with p , and assign the y coordinate of p to be the number of the track in r which is closest to the side of r which is opposite s . For example, if $m = \{6\ 7\ 8\ 12\ 22\ 28\ \dots\}$ is the path which begins with pin b in Figure 3.38, p is pin b , and r is region 6, the s is the bottom side because region 7 is below region 6. Therefore, the y coordinate for pin b is set to track 2.
5. If m contributes a horizontal segment to r , find its position in the ordered list of horizontal path segments for r , and assign the y coordinate of p to be the number of the track in r which corresponds to this position. For example, if $m = \{15\ 10\ 5\ 1\ \dots\}$ is the path which begins with pin e in Figure 3.38, r is region 15, and o is the third position, the y

coordinate for pin e is set to track 4.

Figure 3.38 Tracks and routing columns imposed on an example second-pass region map

The exact set of tracks and columns assigned to each path through each of its constituent regions is determined based on the location of its pins, and its position in the horizontal and vertical orders of path segments in each of its constituent regions. There are essentially five ways that a path can occupy space in a region. They are the following:

1. only a single horizontal track is occupied by the path (see Figure 3.39)
2. only a single vertical routing column is occupied by the path (see Figure 3.40)
3. the path bends 90 degrees in the region; part of one horizontal track and part of one vertical routing column are occupied by the path (see Figure 3.41)
4. the path switches tracks in the region; parts of two horizontal tracks and part of one vertical routing column are occupied by the path (see Figure 3.42)
5. the path switches routing columns in the region; parts of two vertical routing columns and part of one horizontal track are occupied by the path (see Figure 3.43)

Figure 3.39 Horizontal passage through a region

Figure 3.40 Vertical passage through a region

Figure 3.41 90 degree bends in a region

Figure 3.42 Changing tracks in a region

Figure 3.43 Changing routing columns in a region

Once it is known how each path in S_f occupies space in each of its constituent regions (ie. the final track and routing column assignments have been made), the metal routing problem is completely solved.

Chapter 4: TSLR with an Example Leaf Cell Circuit

4.1 The Transmission Gate XNOR Circuit

The transmission gate XNOR circuit of Figure 4.1 can be expressed in terms of two ordered lists of p-type and n-type transistors, S_p and S_n respectively.

$$S_p = \{(AXNORB \text{ AXORB } VDD) (VDD \text{ A-} \text{ A}) (A \text{ B AXORB}) (AXORB \text{ A } B)\}$$

$$S_n = \{(AXNORB \text{ AXORB } VSS) (VSS \text{ A } \text{ A-}) (A \text{ B AXORB}) (AXORB \text{ A-} \text{ B})\}$$

Figure 4.1 Schematic diagram for the transmission gate XNOR circuit

For the purposes of this example, the following assumptions have been made:

1. all p-type transistors are the same size - they have enough room to route three horizontal tracks of metal
2. all n-type transistors are the same size - they have enough room to route two horizontal tracks of metal
3. local interconnect (LI) is used for routing connections
4. the top and bottom horizontal areas of the layout space do not exist

Figure 4.2 contains a pictorial representation of the placement of the 8 transistors of S_p and S_n .

Figure 4.2 Transistor placement

Figure 4.3 depicts the division of the layout space into rectangular regions for the first pass.

Figure 4.3 Region map for the first pass

Figure 4.4 contains the adjacency graph which corresponds to the regions in Figure 4.3. Note that pins associated with two regions in Figure 4.2 have been depicted in Figure 4.4 on the edge between the vertices associated with the two regions, and pins associated with only one region in Figure 4.3 are depicted in Figure 4.4 immediately beside the vertex associated with that one region

Figure 4.4 Adjacency graph for the first pass

There are two additional things to note about the layout space in Figure 4.3 and the corresponding adjacency graph of Figure 4.4. First, region 1 occupies space on both sides of interval boundary 1. This is because LI can be used to connect the source diffusions of the first transistor pair (both the p and n transistors carry the net AXNORB on their source terminals), and there are no other terminals in the circuit which carry the same net so no metal pins are required. Secondly, spacer region 8 exists because the drain net of the second p transistor (A-) is not equal to the source net of the third p transistor (A). Region 8 is not a full spacer because the second and third n transistors do share the common net A-.

4.2 Reduction to a Metall Routing Problem

The nets in the set $H = \{B A- A AXORB\}$ have connections which need to be routed in metal. All connections for the net AXNORB are implemented entirely in LI. The connection of the common diffusion between the first and second p transistors to the VDD wire at the top of the cell is implemented in diffusion. The connection of the common diffusion between the first and second n transistors to the VSS wire at the bottom of the cell

is implemented in diffusion. The following connections of the nets in H are implemented in diffusion:

1. the connection between the drain of the second n transistor and the source of the third n transistor (both carry the net A-)
2. the connection between the drain of the third p transistor and the source of the fourth p transistor (both carry the net AXORB)
3. the connection between the drain of the third n transistor and the source of the fourth n transistor (both carry the net AXORB)

The following connections of the nets in H are implemented in poly:

1. the connection between the gates of the first transistor pair (both the p and n transistors carry the net AXORB on their gate terminals)
2. the connection between the gates of the second transistor pair (both the p and n transistors carry the net A on their gate terminals)
3. the connection between the gates of the third transistor pair (both the p and n transistors carry the net B on their gate terminals)

The following connections of the nets in H are implemented in LI:

1. the connection between the drains of the second transistor pair (both the p and n transistors carry the net A- on their drain terminals)
2. the connection between the common diffusion between the third and fourth transistors pairs (both the common diffusion between the p transistors and the common diffusion between the n transistors carry the net AXORB)
3. the connection between the drains of the fourth transistor pair (both the p and n transistors carry the net B on their drain terminals)

4.3 The Minimum Set of Two-Point Metall Connections

The or-groups of pins for each of the nets in H are as follows:

B: (8), (11 14 15)

A-: (1 3 4), (13)

A: (2), (5 6), (12)

AXORB: (0), (7 9 10)

Therefore, the sub-nets of the minimum set of two-point connections (sub-nets) for these nets given as two member sets (start-pin end-pin), are:

B: (8 11) (8 14) (8 15)

A-: (13 1) (13 3) (13 4)

A: (2 5) (2 6) (12 5) (12 6)

AXORB: (0 7) (0 9) (0 10)}.

4.4 The Set of All First-Graph Paths for the Sub-nets

A total of 146 paths are generated for the 13 sub-nets of Section 4.3. However, due to the practice of terminating paths at diffusion regions, 51 of these paths are redundant. The 95 unique paths are listed below according to net and sub-net.

net AXORB: (32 paths)

- sub-net (0 7):

(12 10 11 9 5 2 1)

(12 10 8 9 5 2 1)

(12 10 8 5 3 0 1)

(12 10 11 9 7 6 3 0 1)

(12 10 8 7 4)

(12 10 11 9 7 4)

(12 10 11 9 5 4)

(12 10 8 9 5 4)

(12 10 8 6 3 4)

(12 13 11 9 5 2 1)

(12 13 11 8 6 3 0 1)

(12 13 11 9 7 6 3 0 1)

(12 13 11 8 7 4)

(12 13 11 9 7 4)

(12 13 11 9 5 4)

(12 13 11 8 6 3 4)

- sub-net (0 9):

(14 9 5 2 1)

(14 9 8 6 3 0 1)

(14 9 7 6 3 0 1)
(14 9 7 4)
(14 9 5 4)
(14 9 8 6 3 4)
(14 13 11 8 6 3 0 1)
(14 13 11 8 7 4)
(14 13 11 8 6 3 4)

- sub-net (0 10):

(13 11 9 5 2 1)
(13 11 8 6 3 0 1)
(13 11 9 7 6 3 0 1)
(13 11 8 7 4)
(13 11 9 7 4)
(13 11 9 5 4)
(13 11 8 6 3 4)

net A-: (21 paths)

- sub-net (13 1):

(6 8 11 13 15)
(6 8 10 12 15)
(6 8 11 13 14 19 17)
(6 8 11 13 12 16 17)
(6 8 10 12 13 14 19 17)
(6 8 10 12 16 17)

- sub-net (13 3):

(8 11 13 15)
(8 10 12 15)
(8 11 13 14 19 17)
(8 11 13 12 16 17)
(8 10 12 13 14 19 17)
(8 10 12 16 17)

- sub-net (13 4):

(9 11 13 15)
(9 14 15)
(9 8 10 12 15)
(9 11 10 12 15)
(9 14 19 17)
(9 8 10 12 16 17)

(9 11 10 12 16 17)

(9 11 13 12 16 17)

(9 14 13 12 16 17)

net A: (29 paths)

- sub-net (2 5):

(10 12 13 14 9 5 4)

(10 11 9 5 4)

(10 8 9 5 4)

(10 8 6 3 4)

(10 8 7)

(10 12 13 14 9 7)

(10 11 9 7)

- sub-net (2 6):

(11 9 5 4)

(8 9 5 4)

(8 6 3 4)

(8 7)

(11 9 7)

- sub-net (12 5):

(10 12 15)

(10 12 13 14 19 17)

(10 12 16 17)

(10 11 13 15)

(10 11 9 14 15)

(10 8 9 14 15)

(10 11 9 14 19 17)

(10 8 9 14 19 17)

(10 11 13 14 19 17)

- sub-net (12 6):

(11 13 15)

(11 9 14 15)

(8 9 14 15)

(11 9 14 19 17)

(11 13 14 19 17)

(11 13 12 16 17)

(8 9 14 19 17)

(8 9 14 13 12 16 17)

net B: (13 paths)

- sub-net (8 11):

(16 12 10 11)
 (16 12 13)
 (16 12 15 14 9 11)
 (16 17 15 14 9 11)
 (16 17 15 13)

- sub-net (8 14):

(17 15 14 9 11)
 (17 15 12 10 11)
 (17 15 13)

- sub-net (8 15):

(19 14 9 11)
 (19 17 15 12 10 11)
 (19 14 15 12 10 11)
 (19 17 15 13)
 (19 14 13)

No equivalent paths were generated so the set of all paths for all sub-nets, Q, is the set of the 95 unique paths listed above.

4.5 Selection of a Solution Set of First-Graph Paths

Since Q contains only 95 paths, the reduction algorithm is not used before the modified branch and bound solution selection algorithm is applied. The solution, S, consists of one path from each of the following five or-groups of sub-nets:

1. (8 11), (8 14), (8 15) - net B
2. (13 1), (13 3), (13 4) - net A-
3. (2, 5), (2 6) - net A
4. (12 5), (12 6) - net A
5. (0 7), (0 9), (0 10) - net AXORB

With the parameters of the branch and bound set to perform a greedy branching scheme (ie. attempting to find a solution - not necessarily an optimum solution)- as

quickly as possible), and with the bounding function equal to the total of the path lengths of the partial solution and the shortest possible lengths for the remaining paths yet to be added to the partial solution, the following paths are chosen for S:

- (14 9 7 4)- from or-group 1
- (9 14 15)- from or-group 2
- (8 6 3 4)- from or-group 3
- (10 12 15)- from or-group 4
- (17 15 13)- from or-group 5

Three of the paths in S, (9 14 15), (10 12 15) and (17 15 13) must be routed through region 15 so the centre horizontal area is assigned three horizontal tracks for routing. Recall that the pdiff area has three tracks available and the ndiff area has 2 tracks available. Therefore, the total number of horizontal tracks available for this solution set of paths is $3 + 3 + 2 = 8$.

4.6 The Set of All Second-Graph Paths for the Sub-nets

The pins used by the paths in S are 0, 2, 4, 5, 6, 8, 9, 12, 13, and 14. Figure 4.5 depicts the division of the layout space into rectangular regions for the second pass.

Figure 4.5 Region map for the second pass

Figure 4.6 contains the adjacency graph which corresponds to the regions in Figure

4.5.

Figure 4.6 Adjacency graph for the second pass

Table 4.1 Region correspondance between first and second pass

1st graph region	2nd graph regions
0	3
1	0, 1, 2
2	4
3	9
4	10
5	11
6	16
7	17
8	20, 21
9	18, 19
10	22, 23
11	24, 25
12	30
13	31
14	32, 33
15	31
16	39
17	40, 41
18	43, 44, 45
19	42

Table 4.1 corresponds each of the first graph regions with the regions of the second

graph. Note that regions 5-7, 12-15, 26-19, and 34-38 in Figures 4.5 and 4.6 are regions created for the gates of the four transistors, and therefore do not have any corresponding first-graph regions. Also, note that there is only one region on the second graph (region 16 in Figures 4.5 and 4.6) associated with region 6 of Figure 4.3. This is because pin 1 is not used by any of the paths in S . This is also true for the first-graph regions 7, 12, 16 and 19 because pins 3, 7, 11 and 15 respectively, are not used by any path in S . Finally, note that regions 13 and 15 have been merged into the single region 31 on the second graph. Again, this is because pin 10 is not used by any path in S .

A total of 42 unique second-graph paths are generated for the five first-graph paths in S . None of these paths are equivalent, so the set of all second-graph paths for the paths in S , Q_f , contains all 42 paths below, listed according to the first-graph paths for which they were generated:

path (14 9 7 4) for sub-net (0 9): (16 paths)

(33 29 18 17 14 10 6)
 (32 29 18 17 14 10 6)
 (33 29 19 18 17 14 10 6)
 (32 29 19 18 17 14 10 6)
 (33 29 18 17 13 10 6)
 (32 29 18 17 13 10 6)
 (33 29 19 18 17 13 10 6)
 (32 29 19 18 17 13 10 6)
 (33 29 18 17 14 10 7)
 (32 29 18 17 14 10 7)
 (33 29 19 18 17 14 10 7)
 (32 29 19 18 17 14 10 7)
 (33 29 18 17 13 10 7)
 (32 29 18 17 13 10 7)
 (33 29 19 18 17 13 10 7)
 (32 29 19 18 17 13 10 7)

path (9 14 15) for sub-net (13 4): (8 paths)

(19 29 32 31 36)
 (18 29 32 31 36)

(19 29 33 32 31 36)
 (18 29 33 32 31 36)
 (19 29 32 31 37)
 (18 29 32 31 37)
 (19 29 33 32 31 37)
 (18 29 33 32 31 37)

path (8 6 3 4) for sub-net (2 6): (6 paths)

(25 21 20 16 12 9 10 13)
 (24 21 20 16 12 9 10 13)
 (25 21 20 16 12 13)
 (24 21 20 16 12 13)
 (25 21 20 16 12 9 10 14)
 (24 21 20 16 12 9 10 14)

path (10 12 15) for sub-net (12 5): (4 paths)

(23 26 30 31 35)
 (22 26 30 31 35)
 (23 26 30 31 36)
 (22 26 30 31 36)

path (17 15 13) for sub-net (8 14): (8 paths)

(41 37 31 27)
 (40 36 31 27)
 (41 36 31 27)
 (40 36 31 27)
 (41 37 31 28)
 (40 37 31 28)
 (41 36 31 28)
 (40 36 31 28)

4.7 Selection of the Final Solution Set of Paths

Since Q_i contains only 42 paths, it is not necessary to apply the reduction algorithm. The modified branch and bound algorithm is used directly to select a final solution set of paths. With the parameters of the branch and bound again set to perform a greedy branching scheme, and the bounding function set as the length of the paths, the following paths are chosen for the final solution set of paths, S_f :

path (33 29 19 18 17 13 10 7) for sub-net (0 9)

path (18 29 32 31 37) for sub-net (13 4)

path (24 21 20 16 12 13) for sub-net (2 6)

path (22 26 30 31 35) for sub-net (12 5)

path (41 36 31 27) for sub-net (8 14)

4.8 Symbolic Layout Representation of the Solution

Figure 4.7 contains the routing space of Figure 4.6 with the horizontal tracks and vertical routing columns of the symbolic layout grid imposed on top..

Figure 4.7 Imposition of tracks and routing columns on the layout space

Table 4.2 contains the horizontal and vertical segments of each of the paths in S_f .

Table 4.2 Horizontal and vertical solution path segments

S_f path	vertical segments	horizontal segments
(33 29 19 18 17 13 10 7)	(19 18 17)	(33 29 19), (17 13 10 7)
(18 29 32 31 37)	(32 31)	(18 29 32), (31 37)
(24 21 20 16 12 13)	(21 20), (12 13)	(24 21), (20 16 12)
(22 26 30 31 35)	(30 31)	(22 26 30), (31 35)
(41 36 31 27)	-	(41 36 31 27)

The pins used by the paths in S_f are assigned to the tracks and routing columns indi-

cated in Table 4.3.

In Figure 4.8, the metal wires for each path in S_f are shown imposed on the space of Figure 4.7. Notice the following:

- path (33 29 19 18 17 13 10 7) bends 90 degrees in regions 19 and 17, and switches tracks in region 10 (see Figure 4.8)
- path (18 29 32 31 37) bends 90 degrees in regions 32 and 31 (see Figure 4.8)
- path (24 21 20 16 13 12) bends 90 degrees in regions 21, 20 and 12 (see Figure 4.8)
- path (22 26 30 31 35) switches tracks in region 30 and bends 90 degrees in regions 30 and 31 (see Figure 4.8)

Table 4.3 Pin coordinates

pin	track	routing column
0	2	2
2	4	5
4	1	8
5	7	8
6	2	8
8	3	9
9	0	10
12	4	12
13	2	12
14	3	13

Figure 4.8 Metal solution wires

Figure 4.9 contains a plot of the final layout after compaction

Figure 4.9 Compacted plot of the TSLR solution

Chapter 5: Experimental Results

In leaf cell synthesis, routing takes place after transistor placement and sizing is complete (see Figure 5.1). Therefore, the experimental results obtained with the prototype topological router started with a given placement for each of the leaf cell circuits; all p-type transistors are assumed to be of a width capable of accomodating three metall tracks, and all n-type transistors are assumed to be of a width capable of accomodating two metall tracks. The prototype topological router is written in Common LISP plus the object-oriented flavors package. All tests for which the results are displayed in this chapter were run on a SUN SPARC2 workstation.

Figure 5.1 Leaf cell synthesis: flow of operation

Many of the computation times included in this chapter are highly variable and unacceptably long. They have been included here only to give an indication of where further work is required to make this topological single-layer approach to leaf cell routing practical.

5.1 Test Circuits

Four different benchmark circuits were used to test the various phases of the prototype software. They are the following:

1. a 2-input transmission gate XNOR [12] containing a total of 8 transistors
2. a non-dual full adder containing four non-dual gates and two inverters for a total of 24 transistors
3. a JK-type flip flop containing three complex NAND gates, two transmission gates and two inverters for a total of 28 transistors
4. a master-slave D-type flip flop with asynchronous set and reset containing four complex NAND gates for a total of 32 transistors

In each circuit, it is assumed that all p transistors are the same size (capable of accommodating three horizontal metal routing tracks), and all n transistors are the same size (capable of accommodating two horizontal metal routing tracks). No other restrictions are placed on the vertical pitch of the cells, and no restrictions are placed on the L/R extents of the cells. Figures 5.2 through 5.5 contain placement diagrams of each of these four circuits.

$$S_p = \{(AXNORB \ AXORB \ VDD) \ (VDD \ A \ A-) \ (A \ B \ AXORB) \ (AXORB \ A \ B)\}$$

$$S_n = \{(AXNORB \ AXORB \ VSS) \ (VSS \ A \ A-) \ (A- \ B \ AXORB) \ (AXORB \ A- \ B)\}$$

Figure 5.2 Transmission gate XNOR placement

$S_p = \{ (VDD\ B\ N1), (N1\ A\ VDD), (VDD\ C\ N1), (N1\ CARRY\ SUM), (SUM\ C\ N3), (N3\ B\ N2), (N2\ A\ VDD), (VDD\ B\ N8), (N8\ A\ CARRY), (CARRY\ C\ N7), (N7\ A\ VDD), (VDD\ B\ N7), (VSS\ B\ N4), (N4\ A\ VSS), (VSS\ C\ N4), (N4\ CARRY\ SUM), (SUM\ C\ N5), (N5\ B\ N6), (N6\ A\ VSS), (VSS\ B\ N10), (N10\ A\ CARRY), (CARRY\ C\ N9), (N9\ A\ VSS), (VSS\ B\ N9) \}$

Figure 5.3 Non-dual full adder placement

$S_p = \{ (W\ CK\ X), (X\ Y\ VDD), (VDD\ X\ Y), (Y\ R\ VDD), (Y\ CK\ Z), (VDD\ J\ V), (VDD\ Q\ S9634), (S9634\ K\ W), (W\ V\ S9635), (S9635\ Q\ Q), (Q\ Q\ VDD), (VDD\ Q\ Z), (Z\ R\ VDD), (VDD\ Z\ Q) \}$
 $S_n = \{ (W\ CK\ X), (X\ Y\ VSS), (VSS\ X\ S9647), (S9647\ R\ Y), (Y\ CK\ Z), (VSS\ J\ V), (S9636\ Q\ VSS), (VSS\ K\ S9636), (S9636\ V\ W), (W\ Q\ S9636), (Q\ Q\ VSS), (VSS\ Q\ S9651), (S9651\ R\ Z), (VSS\ Z\ Q) \}$

Figure 5.4 JK flip flop placement

$S_p = \{ (S158\ CLK\ VDD), (VDD\ D\ S158), (S158\ S18\ S24), (S24\ RSET\ VDD), (S176\ D\ VDD), (VDD\ CLK\ S176), (S176\ S24\ S18), (S18\ SET\ VDD), (VDD\ CLK\ S194), (S194\ S24\ Q), (Q\ SET\ VDD), (VDD\ Q\ Q), (VDD\ S18\ S213), (S213\ CLK\ Q), (Q\ Q\ VDD), (VDD\ RSET\ Q) \}$
 $S_n = \{ (S160\ CLK\ S155), (S155\ D\ VSS), (VSS\ S18\ S160), (S160\ RSET\ S24), (S178\ D\ S173), (S173\ CLK\ VSS), (VSS\ S24\ S178), (S178\ SET\ S18), (S198\ CLK\ VSS), (VSS\ S24\ S198), (S198\ SET\ S197), (S197\ Q\ Q), (S217\ S18\ VSS), (VSS\ CLK\ S217), (S217\ Q\ S216), (S216\ RSET\ Q) \}$

Figure 5.5 Master-slave D flip flop placement

5.2 First Pass Adjacency Graph

5.2.1 Case 1: LI is used for routing, but no routing is permitted in the top and bottom track areas.

Figures 5.6 through 5.9 contain the region maps and corresponding adjacency graphs for the four test circuits when LI connections are used, but routing is not permitted in the top and bottom track areas. All four test circuits use LI for p-diffusion to n-diffusion connections (eg. LI connects the sources of transistor pair 0 in the XNOR circuit - see Figure 5.6). The JK flip flop circuit uses LI to connect a diffusion output to a poly input (ie. LI connects the sources of transistor pair 10 to the gates of transistor pair 9 - see Figure 5.8). All four test circuits use LI to extend diffusion terminals into the centre track area (eg. LI extends the drain of the p transistor of pair 7 of the master-slave D flip flop into the centre track area - see Figure 5.9).

Figure 5.6 Case 1 region map and adjacency graph for the transmission gate XNOR

Figure 5.7 Case 1 region map and adjacency graph for the non-dual full adder

Figure 5.8 Case 1 region map and adjacency graph for the JK flip flop

• = virtual pin

x = virtual pin

Figure 5.9 Case 1 region map and adjacency graph for the master-slave D flip flop

Table 5.1 Case 1 Graph Statistics

test circuit	# trans. pairs	# regions	# pins	# virtual pins	# interval bounds
Transmission gate XNOR	4	20	16	0	12
Non-dual full adder	12	44	31	15	27
JK flip flop	14	66	47	13	36
Master-slave D flip flop	16	70	56	23	38

There are 5 distinct nets in the XNOR circuit, 17 in the non-dual full adder, 18 in the JK flip flop, and 22 in the master-slave D flip flop. They are as follows:

- XNOR: {AXNORB AXORB A A- B VDD VSS}
- Non-dual full adder: {A B C CARRY- N1 N4 N7 N9 SUM- N3 N2 N5 N6 N8 N10 VDD VSS}
- JK flip flop: {W X Y Q Q- CK CK- R- V J K S9634 S9635 S9636 S9647 S9651 VDD VSS}
- Master-slave D flip flop: {S158 CLK S18 S24 RSET- SET- D- D S176 S194 Q Q- S213 S160 S178 S173 S198 S197 S217 S216 VDD VSS}

In the XNOR circuit, the VDD and VSS nets are routed entirely in diffusion, and the AXNORB net is routed entirely in LI. Therefore, there are only four nets (AXORB, A, B and A-) which require metall routing. The VDD, VSS, N2, N3, N5, N6, N8 and N10 nets of the non-dual full adder circuit are all routed entirely in diffusion, and the SUM- net is routed entirely in LI. Therefore, there are eight nets (A, B, C, CARRY-, N1, N4, N7 and N9) which require metall routing. The VDD, VSS, S9651, S9647, S9635 and S9634 nets of the JK flip flop are routed entirely in diffusion, the J and K nets are routed entirely in poly, and the Q- net is routed entirely in a combination of LI and poly. Therefore, there are nine nets (W, X, Y, Q, CK, CK-, R-, V and S9636) which require metall routing. The VDD, VSS, S216, S197, S173, S213 and S194 nets of the master-slave D flip flop are routed entirely in diffusion, and the D and D- nets are routed entirely in poly. Therefore, there are twelve nets (S158, CLK, S18, S24, RSET-, SET-, Q, Q-, S160, S178, S198 and S217) which require metall routing.

Table 5.2 contains statistics on the number of sub-nets required to completely specify all metal nets of each test circuit. Refer to Appendix C for lists showing all the sub-nets for each of these circuits grouped in or-groups.

Table 5.2 Case 1 Sub-net Statistics

test circuit	# metal nets	# sub-nets	average # sub-nets / net
Transmission gate XNOR	4	13	3
Non-dual full adder	8	55	7
JK flip flop	10	77	8
Master-slave D flip flop	13	110	8

51 ms of CPU time was required to translate the placement information for the XNOR circuit into the adjacency graph of Figure 5.6. The non-dual full adder circuit required 66 ms, the JK flip flop circuit required 117 ms and the master-slave D flip flop circuit required 183 ms to translate the placement information into the adjacency graphs of Figures 5.7, 5.8 and 5.9 respectively.

5.2.2 Case 2: LI is used for routing, and routing is permitted in the top and bottom track areas.

Figures 5.10 through 5.13 contain the region maps and corresponding adjacency graphs for the four test circuits when LI connections are used, and routing is permitted in the top and bottom track areas. All four test circuits use LI to extend diffusion terminals into the top and bottom track areas (eg. LI extends the drains of transistor pair 8 of the non-dual full adder circuit into the top and bottom track areas). In the JK flip flop circuit, LI is used to connect the drain, source and drain terminals of the transistors of the seventh, eighth and ninth transistor pairs

Figure 5.10 Case 2 region map and adjacency graph for the transmission gate XNOR

Table 5.3 Case 2 Graph Statistics

test circuit	# trans. pairs	# regions	# pins	# virtual pins	# interval bounds
Transmission gate XNOR	4	34	31	0	12
Non-dual full adder	12	78	65	25	27
JK flip flop	14	112	89	16	36
Master-slave D flip flop	16	119	106	38	38

As in case 1, some of the nets of each of the test circuits are routed entirely in layers other than metal1. Therefore, only the following nets require metal1 routing in each of the circuits:

- XNOR: 4 nets
AXORB, A, A- and B
- Non-dual full adder: 8 nets
A, B, C, CARRY-, N1, N4, N7 and N9
- JK flip flop: 10 nets
W, X, CK, CK-, Y, V, R-, Z, Q and S9636
- Master-slave D flip flop: 12 nets
S158, S160, CLK, S18, S24, RSET-, SET-, S198, S176, S178, Q, Q- and S217

Table 5.4 contains statistics on the number of sub-nets required to completely specify all metal1 nets of each test circuit. Refer to Appendix C for lists showing all the sub-nets for each of these circuits grouped in or-groups.

Notice that pins in the top track area do not form sub-nets with pins in the ndiff or bottom track areas (eg. in the non-dual full adder circuit, net B's pins 0,2 and 4 belong to an or-group which forms sub-nets with the or-group containing pins 65 through 69 - see section C.2 of Appendix C, but since pin 0 is a top track pin, there are no sub-nets (0 68) or (0 69) because pin 68 is an ndiff pin, and pin 69 is bottom track pin - see Figure 5.11), pins in the bottom track area do not form sub-nets with pins in the pdiff or top track areas (eg. in the non-dual full adder circuit, there are no sub-nets (4 65) or (4 66) - see section C.2 of Appendix C - because pin 4 is bottom track pin, and pin 65 is a top track pin and pin 66 is a pdiff pin - see Figure 5.11), pins in the pdiff area do not form sub-nets with pins in the bottom track area (eg. in the non-dual full adder circuit, net B's pins 85 through 89 belong to an or-group which forms sub-nets with the or-group containing pins 37,39, 41, 42 and 43 - see section C.2 of Appendix C, but since pin 86 is a pdiff pin, there is no sub-net (86 43) because pin 43 is a bottom track pin - see Figure 5.11), and pins in the ndiff area

do not form sub-nets with pins in the top track area (eg. in the non-dual full adder circuit, there is no sub-net (88 41) - see section C.2 of Appendix C - because pin 88 is an ndiff pin and pin 41 is a top track pin - see Figure 5.11).

Table 5.4 Case 2 Sub-net Statistics

test circuit	# metall nets	# sub-nets	average # sub-nets / net
Transmission gate XNOR	4	48	12
Non-dual full adder	8	182	23
JK flip flop	10	178	18
Master-slave D flip flop	13	304	23

67 ms of CPU time was required to translate the placement information for the XNOR circuit into the adjacency graph of Figure 5.10. The non-dual full adder circuit required 150 ms, the JK flip flop circuit required 233 ms and the master-slave D flip flop circuit required 266 ms to translate the placement information into the adjacency graphs of Figures 5.11, 5.12 and 5.13 respectively.

Figure 5.11 Case 2 region map and adjacency graph for the non-dual full adder

Figure 5.12 Case 2 region map and adjacency graph for the JK flip flop

Figure 5.13 Case 2 region map and adjacency graph for the master-slave D flip flop

5.2.3 Case 3: LI is not used for routing, and no routing is permitted in the top and bottom track areas.

Figure 5.14 contains the region map and corresponding adjacency graph for the XNOR circuits when LI connections are not used, and routing is not permitted in the top and bottom track areas.

cuit. Refer to Appendix C for a list showing all the sub-nets for the XNOR circuit grouped in or-groups.

Table 5.6 Case 3 Sub-net Statistics

test circuit	# metal1 nets	# sub-nets	average # sub-nets / net
Transmission gate XNOR	5	13	2

49 ms of CPU time was required to translate the placement information for the XNOR circuit into the adjacency graph of Figure 5.14.

5.2.4 Case 4: LI is not used for routing, but routing is permitted in the top and bottom track areas.

Figure 5.15 contains the region map and corresponding adjacency graph for the XNOR circuits when LI connections are not used, but routing is permitted in the top and bottom track areas.

Table 5.7 Case 4 Graph Statistics

test circuit	# trans. pairs	# regions	# pins	# virtual pins	# interval bounds
Transmission gate XNOR	4	27	22	0	12

As in case 3, the VDD and VSS nets are routed entirely in diffusion. Therefore, there are five nets (AXNORB, AXORB, A, B and A-) which require metal1 routing. Table 5.8 contains statistics on the number of sub-nets required to completely specify all metal1 nets of this test circuit.

Figure 5.15 Case 4 region map and adjacency graph for the transmission gate XNOR

Table 5.8 Case 4 Sub-net Statistics

test circuit	# metall nets	# sub-nets	average # sub-nets / net
Transmission gate XNOR	5	25	5

Refer to Appendix C for a list showing all the sub-nets for the XNOR circuit grouped in or-groups.

50 ms of CPU time were required to translate the placement information for the XNOR circuit into the adjacency graph of Figure 5.15.

5.3 Path Generation for the First Pass

Table 5.9 First Pass Path Statistics

test circuit	# paths generated	# unique paths	avg # paths / sub-net	path generation time (ms)	path storage time (ms)
Transmission gate XNOR w/o LI (no routing in top and bottom areas)	57	45	11	167	33
Transmission gate XNOR w/ LI (no routing in top and bottom areas)	146	95	7	266	67
Transmission gate XNOR w/o LI (routing in top and bottom areas)	181	169	7	517	117
Transmission gate XNOR w/ LI (routing in top and bottom areas)	486	347	7	1566	200
Non-dual full adder w/ LI (no routing in top and bottom areas)	612	537	10	1433	450
Non-dual full adder w/ LI (routing in top and bottom areas)	2014	1862	11	9900	1567
JK flip flop w/ LI (no routing in top and bottom areas)	1889	1745	23	7883	1950
JK flip flop w/ LI (routing in top and bottom areas)	4731	4436	25	26984	6383
Master-slave D flip flop w/ LI (no routing in top and bottom areas)	1914	1800	16	10378	1900
Master-slave D flip flop w/ LI (routing in top and bottom areas)	5697	5271	17	35783	7250

Due to the early termination of paths at the nearest diffusion region, some redundant paths are generated for each circuit. The third column of Table 5.9 indicates the number of generated paths which are unique. The last column in the table indicates the times to store the unique paths in the hash table and to build the group trees. All times given in the table are given in units of CPU ms.

5.4 Reduction Algorithm

5.4.1 Calculating Costs for the Paths

The termination criteria for the reduction algorithm depends on the extremes (ie. maximum and minimum) in path length and number of crossing paths. Table 5.10 contains averages of these statistics for the sub-nets of each of the non-dual full adder and the two flip flop test circuits. The path sets for the XNOR test circuit were not large enough to warrant reduction.

Table 5.10 Cost Statistics for First Pass Paths

test circuit	avg min path length	avg max path length	avg min # crossing paths	avg max # crossing paths
Non-dual full adder w/ LI (no routing in top and bottom areas)	4	7	18	62
Non-dual full adder w/ LI (routing in top and bottom areas)	4	9	34	143
JK flip flop w/ LI (no routing in top and bottom areas)	4	9	30	258
JK flip flop w/ LI (routing in top and bottom areas)	4	9	70	419
Master-slave D flip flop w/ LI (no routing in top and bottom areas)	4	8	39	221
Master-slave D flip flop w/ LI (routing in top and bottom areas)	4	9	79	504

5.4.2 Reducing the Path Sets

Table 5.11 contains statistics on the paths for five of the test circuits before and after application of the reduction heuristic. All times in the table are given in units of CPU s/

Table 5.11 Reduction Heuristic Statistics

test circuit	# original paths	# paths after reduction	reduction time (s)
Non-dual full adder w/ LI (no routing in top and bottom areas)	537	440	78.5
Non-dual full adder w/ LI (routing in top and bottom areas)	1862	1586	779.9
JK flip flop w/ LI (no routing in top and bottom areas)	1745	942	1420.9
Master-slave D flip flop w/ LI (no routing in top and bottom areas)	1800	1278	1081.0
Master-slave D flip flop w/ LI (routing in top and bottom areas)	5271	3538	12379.8

5.4.3 Eliminating Virtual Pins

Table 5.12 contains statistics on the long sub-nets produced as a result of eliminating virtual pins in four of the test circuits. Note that the last column in the table lists the number of *unique* long sub-net paths. A number of redundant long sub-net paths are generated due to the removal of open and closed loops during the merging process. All times in the table are given in units of CPU s.

Table 5.12 Long Net Statistics

test circuit	# virtual pin sub-nets	# virtual pin paths	# long sub-nets	# unique long paths	merging time (s)
Non-dual full adder w/ LI (no routing in top and bottom areas)	42	749	7	1666	7.7
JK flip flop w/ LI (no routing in top and bottom areas)	41	4240	11	14064	1388.0
Master-slave D flip flop w/ LI (no routing in top and bottom areas)	55	2326	7	9232	494.7
Master-slave D flip flop w/ LI (routing in top and bottom areas)	151	15272	35	53154	12307.2

Refer to Appendix D for a detailed description of which virtual-pin sub-nets combine to form the long sub-nets for each of the four test circuits of Table 5.12.

5.5 Branch and Bound Solution for the First Pass

Branch and bound solution selection was tested on the four variations of the XNOR circuit. In all cases, the greedy search strategy of terminating upon discovery of the first complete solution was employed. The branch/dive control parameters DELTA and INCR were set to 4.0 and 40 respectively for the two XNOR variations which did not use LI for routing, and to 4.0 and 40000 for the two XNOR variations which used LI for routing. The value of 40000 for INCR was used to encourage diving and thus reduce the solution time. The times in Table 5.13 are again given in units of CPU ms.

Table 5.13 First Pass Branch and Bound Statistics

test circuit	# paths in solution	# nodes searched	# dives & branches	solution time (ms)
Transmission gate XNOR w/o LI (no routing in top and bottom areas)	9	167	113 & 54	3583
Transmission gate XNOR w/ LI (no routing in top and bottom areas)	5	5	5 & 0	667
Transmission gate XNOR w/o LI (routing in top and bottom areas)	9	9	0 & 9	567
Transmission gate XNOR w/ LI (routing in top and bottom areas)	5	5	5 & 0	2933

The following first-pass solution paths found for these circuits are sketched out in Figures 5.16 through 5.19:

1. XNOR w/o LI (no routing in top and bottom areas):

(15 14 13), (13 14 11) - B
 (9 10 11), (9 7) - A
 (8 5 4 3 6), (6 7 9 10 13 14) - A-
 (12 8 7 9 10), (10 9 7 6 3 0 1) - AXORB
 (2 1 0) - AXNORB

2. XNOR w/o LI (routing in top and bottom areas)

(25 24 23), (23 18 13) - B
 (13 12 11), (11 12 15 17 22 24) - A-
 (16 15 17), (16 12 13 14) - A

(20 19 18), (18 16 12 11 6 1 2) - AXORB

(3 2 1) - AXNORB

Note that when LI is not used for routing, it becomes necessary to split some of the diffusion pins in order to complete the connections (eg. pin 6 in Figure 5.16). It is also necessary to extend a poly wire horizontally in one case. (eg pin 19 in Figure 5.17).

Figure 5.16 First pass solution for XNOR w/o LI, no routing in top and bottom areas

Figure 5.17 First pass solution for XNOR w/o LI, routing in top and bottom areas

3. XNOR w/ LI (no routing in top and bottom areas):

(17 15 13) - B

(10 12 15), (8 6 3 4) - A

(9 11 13 15) - A-

(14 9 7 4) - AXORB

Figure 5.18 First pass solution for XNOR w/ LI, no routing in top and bottom areas

4. XNOR w/ LI, routing in top and bottom areas:

(32 23 24) - B

(19 24 27) - A-

(16 20 25), (18 14 8 9) - A

(23 14 15 9) - AXORB

Figure 5.19 First pass solution for XNOR w/ LI, routing in top and bottom areas

5.6 Second Pass Adjacency Graph

Figures 5.20 through 5.23 contain the second-pass region maps and corresponding adjacency graphs for the four variations of the XNOR circuit..

Figure 5.20 Second pass region map and adjacency graph for XNOR w/o LI, no routing in top and bottom areas

Figure 5.21 Second pass region map and adjacency graph for XNOR w/o LI, routing in top and bottom areas

Figure 5.22 Second pass region map and adjacency graph for XNOR w/ LI, no routing in top and bottom areas

Figure 5.23 Second pass region map and adjacency graph for XNOR w/ Li, routing in top and bottom areas

Table 5.14 Second Pass Graph Statistics

test circuit	# required tracks	# regions	# pins used	# interval bounds	graph construction time
Transmission gate XNOR w/o LI (no routing in top and bottom areas)	9	47	14	16	83
Transmission gate XNOR w/ LI (no routing in top and bottom areas)	8	46	10	16	67
Transmission gate XNOR w/o LI (routing in top and bottom areas)	12	71	14	16	83
Transmission gate XNOR w/ LI (routing in top and bottom areas)	9	70	10	16	83

All times in Table 5.14 are given in units of CPU ms.

5.7 Path Generation for the Second Pass

Table 5.15 contains the statistics on the second-pass paths generated for each first-pass path in the solution for the four variations of the XNOR circuit. Again, all times in the table are given in units of CPU ms.

Table 5.15 Second Pass Path Statistics

test circuit	# 2nd pass paths	# 1st pass paths	avg paths / 1st pass path	paths generation time
Transmission gate XNOR w/o LI (no routing in top and bottom areas)	564	9	63	39333
Transmission gate XNOR w/ LI (no routing in top and bottom areas)	36	5	7	350
Transmission gate XNOR w/o LI (routing in top and bottom areas)	343	9	38	11867
Transmission gate XNOR w/ LI (routing in top and bottom areas)	23	5	5	267

The resulting path sets for the XNOR test circuits which use LI are small enough that it is not necessary to apply the reduction algorithm. Only these two test circuits are taken through the remaining steps to produce a final compacted plot of the solution.

5.8 Branch and Bound Solution for the Second Pass

Branch and bound solution selection was tested on the two variations of the XNOR circuit which use LI for routing. In both cases, the greedy search strategy of terminating upon discovery of the first complete solution was employed. The branch/dive control parameters DELTA and INCR were set to 4.0 and 40000 respectively. All times in the table are given in units of CPU ms.

Table 5.16 Second Pass Branch and Bound Statistics

test circuit	# paths in solution	# nodes searched	# dives & branches	solution time (ms)
Transmission gate XNOR w/ LI (no routing in top and bottom areas)	5	5	5 & 0	233
Transmission gate XNOR w/ LI (routing in top and bottom areas)	5	6	5 & 1	267

Figures 5.24 and 5.25 contain the plots of the compacted layouts for the two XNOR circuits. The solution set of second-pass paths which correspond to these plots are as follows:

1. XNOR w/ LI (no routing in top and bottom areas):

(32 29 19 18 17 14 10 7) - AXORB
 (18 25 28 31 37) - A-
 (24 21 20 16 12 13) - A
 (22 26 30 31 35) - A
 (40 36 31 28) - B

2. XNOR w/ LI (routing in top and bottom areas):

(49 43 29 30 24 19 14) - AXORB
 (39 29 23 24) - A
 (35 40 46 53) - A
 (31 45 51 58) - A-
 (62 56 50 51 44) - B

Figure 5.24 Plot of the compacted layout for XNOR, no routing in top and bottom area

Figure 5.25 Plot of the compacted layout for XNOR, routing in top and bottom areas

Chapter 6: Conclusions

6.1 Summary

The topological approach to leaf cell routing has its origins in the desire to be able to consider the nets of a circuit in parallel. Such an approach can avoid the pitfalls of routing connections in a manner which prevents the completion of other nets in the circuit (necessitating a rip-up and re-route phase), or which is optimal for the given connection, but results in a sub-optimal solution for the leaf cell circuit as a whole. The feasibility of producing acceptable layouts with such a topological approach to leaf cell routing has been demonstrated with a Common Lisp implementation of the main elements of a prototype router. Many of the computation times for the tests of the prototype router presented Chapter 5 are highly variable and unacceptably large. This will be addressed in the future work outlined in Section 6.3.

The prototype implementation has enabled this study to successfully meet all of its objectives. The methods developed for representing the layout space of a leaf cell as an adjacency graph make it possible to effectively consider in parallel a large number of routes for the nets of a leaf cell circuit. Algorithms have also been developed to generate all practical routes for all nets of a circuit and to store them in a manner which makes such tasks as eliminating redundant routes and determining the validity of a solution more efficient. Finally, the following secondary aspects of the leaf cell routing problem were explored:

1. the conversion of the routing results into a symbolic layout representation for compaction
2. methods for using the local interconnect layer in leaf cell design synthesis

6.2 Contributions

A new representation of the layout space was introduced. This representation allows a (practically) large number of routes for the nets in a leaf cell circuit to be considered in parallel when finding a solution for the leaf cell routing problem. It is possible (though not trivial) to convert a solution in this representation to a symbolic layout representation for compaction and final conversion into fabrication masks for the silicon.

Two new uses for the local interconnect layer in leaf cell design were discovered. First, when routing is permitted in the top (bottom) track area, local interconnect may be used to route one of the diffusion terminals of a folded transistor. Second, local interconnect wires may be used to extend diffusion nets into the centre, top and bottom track areas of the layout space. This can help reduce congestion when many metal connections are routed through the diffusion areas. It may also make it easier to complete a connection to such a diffusion terminal because the "available area" of the terminal has effectively been increased.

6.3 Future Work

Immediate future work on the prototype of the topological router will centre on what must be completed before the current prototype can be considered fully functional. The following five sections outline this short-term work:

6.3.1 Virtual Pins

The virtual pins which are used to temporarily divide up longnets are currently placed manually in the adjacency graph structure. An algorithm for automatically determining the locations of these pins must be developed. This algorithm can take advantage of two things which have been learned from the manually placed virtual pins. First, since virtual pins are removed before a solution is chosen when the long nets are recombined, they can be located in positions which divide the long nets into sections of similar length,

regardless of whether or not a real pin already exists at those positions. Second, observation of the numbers of paths generated for sections of different length has shown that sections which span more than three transistor pairs (approximately eight interval boundaries) are too long.

6.3.2 Mapping to Symbolic Layout

The problem of mapping the solution from the representation used in the prototype topological router to the fixed coordinate system used in symbolic layout representations is not a trivial one. The current algorithm used in the prototype software effectively handles transistor placement, and all connections routed in layers other than metal1. However the algorithms which translate the routes of the metal1 nets need to be expanded and improved. These algorithms include the assignment of the pin locations, the division of each route into horizontal and vertical segments, the ordering of these segments in each region of the adjacency graph, and the final assignment of these segments to horizontal tracks and vertical routing columns in the symbolic layout representation.

6.3.3 I/O Ports

Currently, I/O ports for the external nets of the leaf cell circuits are not implemented on any layer. This feature, although crucial, will be a fairly trivial addition to the prototype software. I/O ports will be automatically generated for all nets in poly (at the top and bottom of the leaf cell) and metal2, and in local interconnect and metal1 (at the sides of the cell) wherever possible.

6.3.4 Effects of Transistor Sizing

The division of the layout space into regions for the first pass of the routing algorithm is designed to handle transistors of different sizes. However, the remaining algorithms currently assume that all transistors of the same type are of the same size. Since transistor sizing is a desired feature in leaf cell synthesis systems, the remaining algorithms need to be expanded to account for the effects of transistor sizing.

6.3.5 Reduction Algorithm for the Second Pass

The reduction algorithm for the first pass of the routing algorithm bases its termination criteria on the lengths and number of crossings of the paths of the nets. However, since the paths generated in the second pass of the algorithm are simply more detailed versions of only the paths chosen for the metal solution, all paths for a given connection will be of similar length and many will have no crossings. In some cases, a sufficiently small number of paths are generated for all the connections of the circuit, and the branch and bound solution selection can be applied directly. However, when the reduction algorithm is required, the first pass termination criteria is not effective, so a different termination criteria for the second pass must be found.

For the longer term, much exploratory work remains. Improving the speed and efficiency of existing algorithms is the first priority, but time will be spent on developing new features which will add to the functionality of the prototype topological router. The following sections outline the current vision of work for the longer term:

6.3.6 Reduction Algorithm

There are two facets of the reduction algorithm to be explored. First, since the reduction algorithm is one of the more time-consuming sections of the topological routing algorithm, methods for speeding up the reduction algorithm will be investigated. One idea for this has already come to mind; it may be possible to use the condition of path overlap in two directions (horizontal and vertical) instead of path crossing. The group tree structure in the current prototype can be used to quickly find paths which overlap in the horizontal direction. Then, only two comparisons need be applied to each horizontally overlapping path to see if the overlap occurs in both directions. Although some of the paths which overlap in both directions will not actually cross, it is significantly faster to determine two-direction overlap and may prove to be an equally useful variable in the termination criteria for the reduction algorithm. The second avenue of exploration will focus

on evaluating other termination criteria and cost functions, mainly for reducing the set of paths generated in the first pass.

6.3.7 Branch and Bound Solution Search

In many circuits, the most time-consuming part of the algorithm is the branch and bound searching for solutions. Therefore, various combinations of the branch and bound parameters will be tested for on a variety of test circuits to find the values of those parameters which give the best trade-off between solution quality and solution time. In addition, there is room for experimentation with the cost function which controls the bounding aspect of the solution search. Much advantage can be gained by developing a cost function which more closely reflects the cost of the best solution for a given branch.

6.3.8 Constrained Layout

The current prototype operates under the premise that as many metal tracks as are required are available. Therefore, the possibility of introducing constraints on the area of the cells will be investigated. Also, since many high-speed, high-performance circuits have timing and noise constraints on some nets, the possibility of introducing such constraints into the prototype routing software will be explored.

Appendices

APPENDIX A: Glossary of Terms

leaf cell	a small circuit which implements a basic logic function (eg. NAND, OR, AOI, etc.). Leaf cell circuits are used to construct larger blocks of combinational logic.
LOD	line-of-diffusion
LI	local interconnect; an extra layer available for routing in some advanced CMOS technologies. In the 0.8 μm biCMOS process introduced by Texas Instruments [14], local interconnect is a very thin layer of Titanium Nitrate (TiN). In Phillips Research Laboratories's 0.5 μm CMOS process [15], local interconnect is a layer of Titanium Silicide (Ti Si_2).
net	set of physical objects that carry the same electrical signal
non-dual	a CMOS circuit where the function implemented with the p transistors is not the logical dual of the function implemented with the n transistors
pin	a metal1-to-poly contact, a metal1-to-local-interconnect contact or a metal1-to-diffusion contact Note: a terminal may have more than one pin associated with it.
region	a rectangular section of the layout space
region map	the set of regions covering the layout space for a leaf cell circuit
sub-net	a section of a net which connects exactly two pins

- terminal** the gate, drain or source of a transistor
- track** a horizontal line in the layout space along which wires are run (If a Cartesian grid is imposed on the layout space, there is a track for each y-coordinate of the grid.)
- TSLR** Topological Single-Layer Routing

APPENDIX B: Examples of the Cases of Crossing Paths

B.1 Crossings Paths With a Single Region in the Common Section

- with s_1 and s_2 both above-right of r , the order is $(s_1 \ s_2 \ e_1 \ e_2)$

path $k_1 = \{10\ 8\ 9\ 7\ 3\ 1\}$

path $k_2 = \{6\ 7\ 12\ 13\}$

- with s_1 and s_2 both above-right of r , the order is $(s_1 e_2 e_1 s_2)$

path $k_1 = \{10\ 8\ 5\ 7\ 3\ 1\}$

path $k_2 = \{8\ 9\ 7\ 5\}$

- with s_1 and e_2 both above-right of r , and e_1 and s_2 both above left of r , the order is $(s_1 e_2 e_1 s_2)$

path $k_1 = \{15\ 10\ 6\ 0\ 2\}$

path $k_2 = \{1\ 6\ 9\ 14\ 15\ 17\}$

- the order is $(s_1\ s_2\ e_1\ e_2)$

path $k_1 = \{17\ 16\ 12\ 9\ 10\ 4\ 2\}$

path $k_2 = \{7\ 9\ 13\ 17\ 21\}$

- the order is $(s_1 s_2 e_1 e_2)$

path $k_1 = \{17\ 16\ 12\ 9\ 10\ 4\ 2\}$ path $k_2 = \{7\ 9\ 14\ 18\ 22\}$

- the order is $(s_1 e_2 e_1 s_2)$

path $k_1 = \{22\ 18\ 15\ 16\ 8\ 3\ 2\}$ path $k_2 = \{17\ 11\ 8\ 7\ 14\ 15\ 20\ 24\ 25\}$

B.7 Crossings Paths With at Least Two Regions in the Common Section

- the order is $(s_1 s_2 e_1 e_2)$

path $k_1 = \{10\ 11\ 9\ 6\ 5\ 2\ 1\ 0\}$

path $k_2 = \{8\ 9\ 11\ 13\ 15\ 14\}$

- with s_1 and s_2 both above-right of s_x , the order is $(s_1 e_2 e_1 s_2)$

path $k_1 = \{23\ 20\ 16\ 13\ 12\ 7\ 2\}$

path $k_2 = \{17\ 13\ 12\ 7\ 6\ 1\ 0\}$

- the order is $(s_1 e_2 e_1 s_2)$

path $k_1 = \{16\ 15\ 11\ 8\ 7\ 2\ 0\}$

path $k_2 = \{12\ 11\ 15\ 20\ 22\ 23\ 24\}$

- with s_1 and s_2 both above-right of s_x , the order is $(s_1 s_2 e_1 e_2)$

path $k = \{21\ 18\ 19\ 15\ 8\ 3\ 2\}$

path $k_2 = \{14\ 15\ 8\ 10\}$

APPENDIX C: Test Circuit Sub-nets (by or-group)

C.1 Case 1: LI is used for routing, but no routing is permitted in the top and bottom track areas.

- **XNOR: 5 or-groups**

AXORB: (0 7) (0 9) (0 10)
 A-: (13 1) (13 3) (13 4)
 A: (2 5) (2 6)
 (12 5) (12 6)
 B: (8 11) (8 14) (8 15)

- **Non-dual full adder: 18 or-groups**

N1: (0 6) (0 9) (3 6) (3 9)
 B: (1 31) (1 32) (1 33)
 (13 31) (13 32) (13 33)
 (13 15)
 (15 34) (15 35) (15 36)
 (27 34) (27 35) (27 36)
 N4: (2 8) (2 10) (4 8) (4 10)
 A: (5 37) (5 38) (5 39)
 (14 37) (14 38) (14 39)
 (14 17)
 (17 25)
 C: (7 12)
 (12 40) (12 41) (12 42)
 (21 40) (21 41) (21 42)
 CARRY-: (11 43) (11 44) (11 45)
 (43 16) (43 18) (43 19) (44 16) (44 18) (44 19)
 (45 16)
 (45 18) (45 19)
 N7: (20 26) (20 28) (23 36) (23 28)
 N9: (22 29) (22 30) (24 29) (24 30)

- **JK flip flop: 21 or-groups**

W: (0 47) (0 48) (1 47) (1 48) (2 47) (2 48)
 (47 49) (48 49)
 (49 27) (49 29)
 (27 32) (27 33) (29 32) (29 33)
 X: (10 3) (10 6) (10 7)
 CK: (4 18)

CK-: (5 17)
 Y: (8 9) (8 11)
 (9 13) (9 15) (9 14) (11 13) (11 15) (11 14)
 R-: (12 50)
 (50 51) (50 52) (50 53)
 (40 51) (40 52) (40 53)
 Z: (16 54) (16 55) (16 56) (19 54) (19 55) (19 56) (20 54) (20 55) (20 56)
 (54 57) (54 58) (54 59) (55 57) (55 58) (55 59) (56 57) (56 58) (56 59)
 (57 37) (57 39) (58 37) (58 39) (59 37) (59 39)
 (37 41) (37 42) (39 41) (39 42)
 V: (31 21) (31 22) (31 23)
 S9636: (24 28) (24 30) (25 28) (25 30)
 (28 34) (28 35) (30 34) (30 35)
 Q: (26 36)
 (38 43) (38 45) (38 46)

• Master-slave D flip flop: 29 or-groups

S158: (0 5) (0 6) (1 5) (1 6)
 S160: (2 9) (2 11) (3 9) (3 11)
 CLK: (4 56) (4 57) (4 58)
 (20 56) (20 57) (20 58)
 (20 32)
 (32 59) (32 60) (32 61)
 (47 59) (47 60) (47 61)
 S24: (7 13) (7 14) (10 13) (10 14)
 (23 13) (23 14)
 (23 34)
 S18: (8 62) (8 63) (8 64)
 (62 22) (62 25) (63 22) (63 25) (64 22) (64 25)
 (22 28) (22 29) (25 28) (25 29)
 (28 65) (28 66) (28 67) (29 65) (29 66) (29 67)
 (45 65) (45 66) (45 67)
 RSET: (12 68) (12 69) (12 70)
 (68 71) (68 72) (69 71) (69 72) (70 71) (70 72)
 (71 73) (71 74) (71 75) (72 73) (72 74) (72 75)
 (73 76) (73 77) (73 78) (74 76) (74 77) (74 78) (75 76) (75 77) (75 78)
 (53 76) (53 77) (53 78)
 S176: (15 19) (15 21) (16 19) (16 21)
 S178: (17 24) (17 26) (18 24) (18 26)
 SET: (27 38)
 S198: (30 35) (30 37) (31 35) (31 37)

Q: (33 39) (33 41) (33 42) (36 39) (36 41) (36 42)
 (51 39) (51 41) (51 42)
 Q-: (40 46) (40 49)
 (46 52) (46 54) (46 55) (49 52) (49 54) (49 55)
 S217: (43 48) (43 50) (44 48) (44 50)

C.2 Case 2: LI is used for routing, and routing is permitted in the top and bottom track areas.

• XNOR:

AXORB: (0 15) (0 19) (0 20) (1 15) (1 17) (1 19) (1 20) (1 21) (2 17) (2 20) (2 21)
 A: (3 10) (3 11) (3 12) (5 10) (5 11) (5 12) (9 12)
 (22 10) (22 11) (22 12) (24 10) (24 11) (24 12)
 A-: (25 4) (25 6) (25 7) (25 8) (25 13) (29 7) (29 8) (29 13)
 B: (14 23) (14 26) (14 27) (16 23) (16 26) (16 27) (16 28) (16 30) (18 27) (18 28) (18 30)

• Non-dual full adder:

B: (0 65) (0 66) (0 67) (2 65) (2 66) (2 67) (2 68) (2 69) (4 67) (4 68) (4 69)
 (27 65) (27 66) (27 67) (28 65) (28 66) (28 67) (28 68) (28 69) (29 67) (29 68) (29 69)
 (27 33) (27 34) (28 33) (28 34) (28 35) (29 34) (29 35)
 (33 70) (33 71) (33 72) (34 70) (34 71) (34 72) (34 73) (34 74) (35 72) (35 73) (35 74)
 (56 70) (56 71) (56 72) (58 70) (58 71) (58 72) (58 73) (58 74) (63 72) (63 73) (63 74)
 A: (9 75) (9 76) (9 77) (10 75) (10 76) (10 77) (10 78) (10 79) (11 77) (11 78) (11 79)
 (30 75) (30 76) (30 77) (31 75) (31 76) (31 77) (31 78) (31 79) (32 77) (32 78) (32 79)
 (30 36) (30 38) (31 36) (31 38) (31 40) (32 38) (32 40)
 (36 53) (36 54) (38 53) (38 54) (38 55) (40 54) (40 55)
 C: (12 24) (12 25) (14 24) (14 25) (14 26) (16 25) (16 26)
 (24 80) (24 81) (24 82) (25 80) (25 81) (25 82) (25 83) (25 84) (26 82) (26 83) (26 84)
 (44 80) (44 81) (44 82) (46 80) (46 81) (46 82) (46 83) (46 84) (48 82) (48 83) (48 84)
 CARRY-: (21 85) (21 86) (21 87) (22 85) (22 86) (22 87) (22 88) (22 89) (23 87) (23 88) (23 89)
 (85 37) (85 41) (85 42) (86 37) (86 39) (86 41) (86 42) (87 37) (87 39) (87 41) (87 42)
 (87 43) (88 37) (88 39) (88 42) (88 43) (89 39) (89 42) (89 43)
 N1: (1 13) (1 17) (1 18) (5 13) (5 17) (5 18) (6 13) (6 17) (6 18)
 N4: (3 15) (3 19) (3 20) (7 15) (7 19) (7 20) (8 15) (8 17) (8 20)
 N7: (45 57) (45 59) (45 60) (49 57) (49 59) (49 60) (50 57) (50 59) (50 60)
 N9: (47 61) (47 62) (47 64) (51 61) (51 62) (51 64)

• JK flip flop:

W: (0 89) (1 89) (1 90) (2 89) (2 90) (2 91) (3 89) (3 90) (3 91) (4 89) (4 90) (4 91)
 (89 92) (90 92) (91 92)
 (92 51) (92 53) (92 54)
 (51 59) (51 61) (53 61) (54 59) (54 61) (54 62)

CK: (5 34) (7 34) (7 38)
 X: (17 6) (17 11) (17 12) (19 6) (19 9) (19 11) (19 12) (19 13) (20 9) (20 12) (20 13)
 CK-: (8 31) (8 33) (10 33)
 Y: (14 18) (14 21) (14 22) (15 18) (15 21) (15 22) (16 22)
 (18 25) (18 29) (18 27) (18 28) (21 29) (21 27) (21 28) (22 25) (22 29) (22 30) (22 27)
 (22 28)
 R-: (23 93) (24 93) (26 93)
 (93 94) (93 95) (93 96) (93 97) (93 98)
 (75 94) (75 97) (76 94) (76 95) (76 96) (76 97) (76 98) (79 94) (79 95) (79 96) (79 97)
 (79 98)
 Z: (32 100) (32 101) (32 102) (32 9103) (35 100) (35 101) (35 102) (36 99) (36 100) (36 101)
 (36 102) (36 103) (37 99) (37 101) (37 102) (37 103) (39 99) (39 102) (39 103)
 (99 104) (100 104) (101 104) (102 104) (103 104)
 (104 70) (104 73) (104 74)
 (70 77) (70 78) (73 77) (74 77) (74 78) (74 80)
 (81 77)
 V: (57 40) (57 41) (57 42) (58 40) (58 41) (58 42) (58 43) (58 44) (60 42) (60 43) (60 44)
 Q: (48 66) (48 67) (49 66) (49 67) (49 68) (50 67) (50 68)
 (69 82) (69 84) (69 85) (71 82) (71 84) (71 85) (71 86) (71 88) (72 85) (72 86) (72 88)
 S9636: (45 52) (45 55) (45 56) (46 52) (46 55) (46 56) (47 52) (47 55) (47 56)
 (52 63) (52 64) (52 65) (55 63) (55 64) (55 65) (56 63) (56 64) (65 65)

• Master-slave D flip flop:

S158: (0 9) (0 10) (0 11) (1 9) (1 10) (1 11) (2 9) (2 10) (2 11)
 S160: (3 15) (3 19) (3 20) (4 15) (4 19) (4 20) (5 15) (5 19) (5 20)
 CLK: (6 106) (6 107) (6 108) (7 106) (7 107) (7 108) (7 109) (7 110) (8 108) (8 109) (8 110)
 (33 106) (33 107) (33 108) (35 106) (35 107) (35 108) (35 109) (35 110) (36 108) (36 109)
 (36 110)
 (33 57) (33 58) (35 57) (35 58) (35 59) (36 58) (36 59)
 (57 112) (57 113) (57 115) (58 111) (58 112) (58 113) (58 114) (58 115) (59 111) (59 113)
 (59 114)
 (86 112) (86 113) (86 115) (88 111) (88 112) (88 113) (88 114) (88 115) (90 111) (90 113)
 (90 114)
 S18: (12 116) (12 117) (12 118) (14 116) (14 117) (14 118) (14 119) (14 102) (16 118) (16 119)
 (16 120)
 (116 40) (116 44) (116 45) (117 40) (117 44) (117 45) (118 40) (118 44) (118 45) (119 40)
 (119 45) (120 45)
 (40 50) (40 51) (44 50) (45 50) (45 51) (45 53)
 (50 121) (50 122) (50 123) (50 124) (50 125) (51 121) (51 122) (51 123) (51 125) (53 121)
 (53 122) (53 123)
 (83 124) (83 125) (84 121) (84 122) (84 123) (84 124) (84 125) (85 121) (85 122)

(85 123)

S24: (13 23) (13 24) (17 23) (18 23) (18 24) (18 26)
 (39 23) (41 23) (41 24) (41 26) (43 23) (43 24) (43 26)
 (39 60) (39 62) (41 60) (41 62) (41 64) (43 62) (43 64)

RSET-: (21 126) (21 127) (21 128) (22 126) (22 127) (22 128) (22 129) (22 130) (25 128) (25 129)
 (25 130)
 (126 131) (126 133) (127 131) (127 133) (128 131) (128 133) (128 132) (129 133)
 (129 132) (130 132) (130 131)
 (131 134) (131 135) (131 136) (131 137) (131 138) (132 134) (132 136) (132 137)
 (133 135) (133 136) (133 138)
 (134 141) (134 142) (134 143) (135 141) (135 139) (135 140) (136 141) (136 142)
 (136 143) (136 139) (136 140) (137 141) (137 142) (137 143) (138 139) (138 140)
 (138 141)
 (98 139) (98 140) (98 141) (100 139) (100 140) (100 141) (100 142) (100 143) (104 141)
 (104 142) (104 143)

S178: (30 42) (30 46) (30 47) (31 42) (31 46) (31 47) (32 42) (32 46) (32 47)

SET-: (48 69) (48 70) (49 69) (49 70) (49 71) (52 70) (52 71)

Q: (61 73) (61 75) (61 76) (61 77) (65 73) (63 75) (63 76) (65 73) (65 75) (65 76) (65 77)
 (65 79)
 (95 73) (95 75) (95 76) (96 73) (96 75) (96 76) (96 77) (96 79) (97 76) (97 77) (97 79)

Q-: (72 87) (72 91) (72 92) (74 87) (74 91) (74 92) (78 92)
 (87 99) (87 101) (87 102) (87 103) (91 99) (91 101) (91 102) (92 99) (92 101) (92 102)
 (92 103) (92 105)

S176: (27 34) (27 37) (27 38) (28 34) (28 37) (28 38) (29 34) (29 37) (29 38)

S198: (54 63) (54 67) (54 68) (55 63) (55 67) (55 68) (56 63) (56 67) (56 68)

S217: (80 89) (80 93) (80 94) (81 89) (81 93) (81 94) (82 89) (82 93) (82 94)

C.3 Case 3: LI is not used for routing, and no routing is permitted in the top and bottom track areas.

AXNORB: (0 1)

AXORB: (2 7) (2 9) and (7 9)

A-: (12 3) (12 5) and (3 5)

A: (4 6) (4 11) and (11 6)

B: (8 10) (8 13) and (10 13)

C.4 Case 4: LI is not used for routing, but routing is permitted in the top and bottom track areas.

AXNORB: (0 1)

AXORB: (2 12) (3 12) (3 14) (4 14) and (12 14)

A: (5 10) (7 10) (9 10) (5 16) (5 18) (7 16) (7 18) and (9 18) and (16 10) (18 10)

A-: (19 6) (19 8) (21 8) and (6 8)

B: (11 17) (13 17) (13 20) (15 20) and (17 20)

APPENDIX D: Test Circuit Long Sub-nets

D.1 Non-dual full adder: routing is not permitted in the top and bottom track areas

The sub-nets which contain virtual pins are merged into long sub-nets as follows
(Refer to Figure 5.7):

- (1 31), (1 32), (1 33), (13 31), (13 32), (13 33) become (1 13)
- (15 34), (15 35), (15 36), (27 34), (27 35), (27 36) become (15 27)
- (5 37), (5 38), (5 39), (14 37), (14 38), (14 39) become (5 14)
- (12 40), (12 41), (12 42), (21 40), (21 41), (21 42) become (12 21)
- (11 43), (11 44), (11 45), (43 16), (44 16), (45 16) become (11 16)
- (11 43), (11 44), (11 45), (43 18), (44 18), (45 18) become (11 18)
- (11 43), (11 44), (11 45), (43 19), (44 19), (45 19) become (11 19)

D.2 JK flip flop: routing is not permitted in the top and bottom track areas

The sub-nets which contain virtual pins are merged into long sub-nets as follows
(Refer to Figure 5.8):

- (0 47), (0 48), (47 49), (48 49), (49 27) become (0 27)
- (0 47), (0 48), (47 49), (48 49), (49 29) become (0 29)
- (1 47), (1 48), (47 49), (48 49), (49 27) become (1 27)
- (1 47), (1 48), (47 49), (48 49), (49 29) become (1 29)
- (2 47), (2 48), (47 49), (48 49), (49 27) become (2 27)
- (2 47), (2 48), (47 49), (48 49), (49 29) become (2 29)
- (12 50), (50 51), (50 52), (50 53), (40 51), (40 52), (40 53) become (12 40)
- (16 54), (16 55), (16 56), (54 57), (55 57), (56 57), (54 58), (55 58), (56 58), (54 59), (55 59), (56 59), (57 37), (58 37), (59 37) become (16 37)
- (16 54), (16 55), (16 56), (54 57), (55 57), (56 57), (54 58), (55 58), (56 58), (54 59), (55 59), (56 59), (57 39), (58 39), (59 39) become (16 39)
- (19 54), (19 55), (19 56), (54 57), (55 57), (56 57), (54 58), (55 58), (56 58), (54 59), (55 59), (56 59), (57 37), (58 37), (59 37) become (19 37)

- (19 54), (19 55), (19 56), (54 57), (55 57), (56 57), (54 58), (55 58), (56 58), (54 59), (55 59), (56 59), (57 39), (58 39), (59 39) become (19 39)

D.3 Master-slave D flip flop: routing is not permitted in the top and bottom track areas

The sub-nets which contain virtual pins are merged into long sub-nets as follows
(Refer to Figure 5.9):

- (4 56), (4 57), (4 58), (20 56), (20 57), (20 58) become (4 20)
- (32 59), (32 60), (32 61), (47 59), (47 60), (47 61) become (32 47)
- (8 62), (8 63), (8 64), (62 22), (63 22), (64 22) become (8 22)
- (8 62), (8 63), (8 64), (62 25), (63 25), (64 25) become (8 25)
- (28 65), (28 66), (28 67), (45 65), (45 66), (45 67) become (28 45)
- (29 65), (29 66), (29 67), (45 65), (45 66), (45 67) become (29 45)
- (12 68), (12 69), (12 70), (68 71), (68 72), (69 71), (69 72), (70 71), (70 72), (71 73), (71 74), (71 75), (72 73), (72 74), (72 75), (73 76), (73 77), (73 78), (74 76), (74 77), (74 778), (75 76), (75 77), (75 78), (53 76), (53 77), (53 78) become (12 53)

D.4 Master-slave D flip flop: routing is permitted in the top and bottom track areas

The sub-nets which contain virtual pins are merged into long sub-nets as follows
(Refer to Figure 5.9):

- (6 106), (6 107), (6 108), (106 33), (107 33), (108 33) become (6 33)
- (6 106), (6 107), (6 108), (106 35), (107 35), (108 35) become (6 35)
- (7 106), (7 107), (7 108), (106 33), (107 33), (108 33) become (7 33)
- (7 106), (7 107), (7 108), (7 109), (7 110), (106 35), (107 35), (108 35), (109 35), (110 35), become (7 35)
- (7 108), (7 109), (7 110), (108 36), (109 36), (110 36), become (7 36)
- (8 108), (8 109), (8 110), (108 35), (109 35), (110 35) become (8 35)
- (8 108), (8 109), (8 110), (108 36), (109 36), (110 36) become (8 36)
- (57 112), (57 113), (57 115), (86 112), (86 113), (86 115) become (57 86)
- (57 112), (57 113), (57 115), (88 112), (88 113), (88 115) become (57 88)
- (58 112), (58 113), (58 115), (86 112), (86 113), (86 115) become (58 86)

- (58 111), (58 112), (58 113), (58 114), (58115), (88 111), (88 112), (88 113), (88 114), (88 115) become (58 88)
- (58 111), (58 113), (58 114), (90 111), (90 113), (90 114) become (58 90)
- (59 111), (59 113), (59 114), (88 111), (88 113), (88 114) become (59 88)
- (59 111), (59 113), (59 114), (90 111), (90 113), (90 114) become (59 90)
- (12 116), (12 117), (12 118), (116 40), (117 40), (118 40) become (12 40)
- (12 116), (12 117), (12 118), (116 44), (117 44), (118 44) become (12 44)
- (12 116), (12 117), (12 118), (116 45), (117 45), (118 45) become (12 45)
- (14 116), (14 117), (14 118), (14 119), (116 40), (117 40), (118 40), (119 40) become (14 40)
- (14 116), (14 117), (14 118), (116 44), (117 44), (118 44) become (14 44)
- (14 116), (14 117), (14 118), (14 119), (14 120), (116 45), (117 45), (118 45), (119 45), (120 45) become (14 45)
- (16 118), (16 119), (16 120), (118 45), (119 45), (120 45) become (16 45)
- (50 124), (50 125), (83 124), (83 125) become (50 83)
- (50 121), (50 122), (50 123), (50 124), (50 125), (84 121), (84 122), (84 123), (84 124), (84 125) become (50 84)
- (50 121), (50 122), (50 123), (85 121), (85 122), (85 123) become (50 85)
- (51 124), (51 125), (83 124), (83 125) become (51 83)
- (51 121), (51 122), (51 123), (51 125), (84 121), (84 122), (84 123), (84 125) become (51 84)
- (51 121), (51 122), (51 123), (85 121), (85 122), (85 123) become (51 85)
- (53 121), (53 122), (53 123), (84 121), (84 122), (84 123) become (53 84)
- (53 121), (53 122), (53 123), (85 121), (85 122), (85 123) become (53 85)
- (21 126), (21 127), (21 128), (126 131), (126 133), (127 131), (127 133), (128 131), (128 133), (128 132), (131 134), (131 135), (131 136), (131 137), (131 138), (134 141), (135 139), (135 140), (135 141), (136 139), (136 140), (136 141), (137 141), (138 139), (138 140), (138 141), (98 139), (98 140), (98 141) become (21 98)
- (21 126), (21 127), (21 128), (126 131), (126 133), (127 131), (127 133), (128 131), (128 133), (128 132), (131 134), (131 135), (131 136), (131 137), (131 138), (132 134), (132 136), (132 137), (133 135), (133 136), (133 138), (134 141), (134 142), (134 143),

(135 139), (135 140), (135 141), (136 139), (136 140), (136 141), (136 142), (136 143), (137 141), (137 142), (137 143), (138 139), (138 140), (138 141), (100 139), (100 140), (100 141), (100 142), (100 143) become (21 100)

- (22 126), (22 127), (22 128), (22 129), (22 130), (126 131), (126 133), (127 131), (127 133), (128 131), (128 133), (128 132), (129 131), (129 133), (129 132), (130 131), (130 132), (131 134), (131 135), (131 136), (131 137), (131 138), (132 134), (132 136), (132 137), (133 135), (133 136), (133 138), (134 141), (135 139), (135 140), (135 141), (136 139), (136 140), (136 141), (137 141), (138 139), (138 140), (138 141), (98 139), (98 140), (98 141) become (22 98)
- (22 126), (22 127), (22 128), (22 129), (22 130), (126 131), (126 133), (127 131), (127 133), (128 131), (128 133), (128 132), (129 131), (129 133), (129 132), (130 131), (130 132), (131 134), (131 135), (131 136), (131 137), (131 138), (132 134), (132 136), (132 137), (133 135), (133 136), (133 138), (134 141), (134 142), (134 143), (135 139), (135 140), (135 141), (136 139), (136 140), (136 141), (136 142), (136 143), (137 141), (137 142), (137 143), (138 139), (138 140), (138 141), (100 139), (100 140), (100 141), (100 142), (100 143) become (22 100)
- (22 126), (22 127), (22 128), (22 129), (22 130), (126 131), (126 133), (127 131), (127 133), (128 131), (128 133), (128 132), (129 131), (129 133), (129 132), (130 131), (130 132), (131 134), (131 135), (131 136), (131 137), (131 138), (132 134), (132 136), (132 137), (133 135), (133 136), (133 138), (134 141), (135 139), (135 140), (135 141), (136 139), (136 140), (136 141), (137 141), (138 139), (138 140), (138 141), (104 139), (104 140), (104 141) become (22 104)
- (25 128), (25 129), (25 130), (128 131), (128 133), (128 132), (129 131), (129 133), (129 132), (130 131), (130 132), (131 134), (131 135), (131 136), (131 137), (131 138), (132 134), (132 136), (132 137), (133 135), (133 136), (133 138), (134 141), (134 142), (134 143), (135 139), (135 140), (135 141), (136 139), (136 140), (136 141), (136 142), (136 143), (137 141), (137 142), (137 143), (138 139), (138 140), (138 141), (100 139), (100 140), (100 141), (100 142), (100 143) become (25 100)
- (25 128), (25 129), (25 130), (128 131), (128 133), (128 132), (129 131), (129 133), (129 132), (130 131), (130 132), (131 134), (131 135), (131 136), (131 137), (131 138), (132 134), (132 136), (132 137), (133 135), (133 136), (133 138), (134 141), (135 139), (135 140), (135 141), (136 139), (136 140), (136 141), (137 141), (138 139), (138 140), (138 141), (104 139), (104 140), (104 141) become (25 104)

References

- [1] Shiraishi, Y., Sakemi, J., "A Permeation Router", IEEE Trans. on Computer-Aided Design, Vol. CAD-6, No. 3, May 1987, pp. 462-471.
- [2] Ong, C. L., Li, J. T., Lo, C. Y., "GENAC: An Automatic Cell Synthesis Tool", Proc of 26th ACM/IEEE Design Automation Conf., 1989, pp. 239-243.
- [3] Bar-Yehuda, R., Feldman, J. A., Pinter, R. Y., Wimer, S., "Depth- First Search and Dynamic Programming Algorithms for Efficient CMOS Cell Generation", IEEE Trans. on Computer-Aided Design, Vol. 8, No. 7, July 1989, pp. 737-743.
- [4] Domic, A., Levitin, S., Phillips, N., Thai, C., Siple, T., Bhavsar, D., Bissell, C., "CLEO: A CMOS Layout Generator", Proc. of International Conf. on Computer Aided Design, 1989, pp. 340- 343.
- [5] Chen, C. C., Chow, S. L., "The Layout Synthesizer: An Automatic Netlist-to-Layout System", Proc. of 26th ACM/IEEE Design Automation Conf., 1989, pp. 232-238.
- [6] Shiraishi, Y., Sakemi, J., Kutsuwada, M., Tsukizoe, A., Satoh, T., "A High Packing Density Module Generator for CMOS Logic Cells", Proc. of 25th Design Automation Conf., 1988, pp. 439-444.
- [7] Wimer, S., Pinter, R. Y., Feldman, J. A., "Optimal Chaining of CMOS Transistors in a Functional Cell", IEEE Trans. on Computer-Aided Design, Vol. CAD-6, No. 5, September 1987, pp. 795-801.
- [8] Poirier, C. J., "Excellerator: Custom CMOS Leaf Cell Layout Generator", IEEE Trans. on Computer-Aided Design, Vol. 8, No. 7, July 1989, pp. 744-755.
- [9] Bachelu, C. R., Lefebvre, M. C., "A Study of the Use of Local Interconnect in CMOS Leaf Cell Design", Proc. of 1992 IEEE International Symposium On Circuits And Systems, Vol. 3, May 1992, pp. 1258-1261.
- [10] Ullman, J. D., Computational Aspects of VLSI, Computer Science Press, 1984.

- [11] Ullman J.D., Computational Aspects of VLSI, Computer Science Press, 1984, pp. 385-387.
- [12] Weste, N.H.E., K. Eshraghian, Principles of CMOS VLSI Design: A Systems Perspective, Addison-Wesley Publishing Inc., 1985, p.317.
- [13] Weste, N.H.E., K. Eshraghian, Principles of CMOS VLSI Design: A Systems Perspective, Addison-Wesley Publishing Inc., 1985, p.314.
- [14] Lineback, J. R., "TI Finds a New Way to Shrink SRAM Cells," Electronics, July 23, 1987, pp. 63-64.
- [15] Verhaar, R. D. J., R. A. Augur, C. N. A. Aussems, L. de Bruin, F. A. M. Op den Buijsch, L. W. M. Dingen, T. C. T. Genus, W. J. M. Haveermans, A. H. Montree, P. A. van der Plas, H. G. Pomp, M. Vertregt, R. de Werdt, N. A. H. Wils, and P. H. Woerlee, "A 25 μm^2 Bulk Full CMOS SRAM Cell Technology With Fully Overlapping Contacts," Digest IEDM, 1990, pp. 473-476.

END

24 | 05 | 93

FIN