1

MICRO

**National Library
of Canada**

Canadian Theses Service

**Ottawa, Canada
K1A 0N4**

**Bibliothèque nationale
du Canada**

Service des thèses canadiennes

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.

CMOS LEAF CELL SYNTHESIS

by

Martin C. Lefebvre, B.A.Sc., M.Eng.

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

Doctor of Philosophy

Ottawa Carleton Institute for Electrical Engineering
Department of Electronics
Faculty of Engineering
Carleton University
Ottawa, Ontario
April 1989

© copyright 1989, Martin Lefebvre

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

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

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

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

ISBN 0-315-51122-2

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

CMOS LEAF CELL SYNTHESIS

submitted by Martin C. Lefebvre in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Prof. Eduard Cerny
External examiner

Prof. J.S. Wight
Chairman,
Department of Electronics

Prof. C.H. Chan
Thesis Supervisor

Carleton University
April, 1989

ABSTRACT

Leaf cell synthesis consists of automatically generating layout geometries for small circuits at the leaf level of some hierarchical circuit representation. The most important metrics used to evaluate cell synthesis tools include design correctness, area efficiency and the ability to effectively handle physical constraints, such as design rules, target cell structure and transistor sizes. This thesis addresses all the critical issues associated with each phase of the cell synthesis process: transistor level design, device level placement, routing of internal nets and symbolic layout compaction. It describes a prototype leaf cell synthesis system, called Picasso, capable of mapping a gate level description into a complete set of layout masks, based on the line of diffusion layout style.

Picasso's method for optimizing the layout of CMOS complex gates imposes fewer restrictions than other approaches on the nature of layouts produced. It also minimizes the amount of internal interconnect, in addition to forming optimal chains of transistors. The transistor placement algorithm uses a composite metric which applies connectivity and optimal chaining considerations simultaneously. The routing algorithm is capable of routing over transistor chains and makes efficient use of residual routing areas resulting from unequal transistor sizes. Finally, Picasso introduces a new symbolic layout compaction method, based on the definition of forbidden zones, associated with each layer, and represented as "flexible boundaries". The final shape of layout objects is determined during the compaction phase, using boundary definitions as a guide, which eliminates the need for the explicit insertion of jogs in the symbolic layout. Furthermore, non-Mahattan geometries are handled naturally by this new method. Flexible boundaries are an extension of "contour-based" compaction methods presented earlier in the contexts of optimal river routing and channeled compaction.

ACKNOWLEDGEMENTS

The author wishes to thank his fellow graduate students and staff of the Department of Electronics for creating a superb work environment. Special thanks to my advisor, Prof. C.H. Chan, for his support throughout my stay at Carleton University.

A number of people have given me assistance on a technical level for which I wish to express my gratitude. Prof. D. Al-Khalili has been of particular help, in the early going, by providing the human cell design expertise that I have strived to emulate with a computer program. Dr. John Hogeboom and Mr. Go Sunatori of Bell-Northern Research have provided me with much appreciated feedback and advice.

My participation in a Ph.D. program has been made possible by my employer, Bell-Northern Research, whose post-graduate scholarship program represents a great deal more than financial assistance for participants. In particular, many thanks to the program administrators the late Dr. R.R. Jackson and Dr. J. Dymont, to my former manager Youssef Geadah, and my present manager Grant Martin for making it possible for me to complete my work on a part time basis.

Finally, I wish to acknowledge the financial participation of NSERC through a 1967 Science and Engineering Scholarship, of the Canadian Microelectronics Corporation, who provided the SUN/3 workstation used for prototype implementation, of Carleton University and of course Bell-Northern Research.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	x
LIST OF APPENDICES	xiii
 Chapter 1: INTRODUCTION	 1
1.1 Objectives	1
1.2 Motivation	2
1.2.1 Impact on ASIC Design Flow	2
1.2.2 Application to Logic Synthesis	3
1.2.3 Impact on Design Portability	5
1.3 Outline of the Thesis	6
 Chapter 2: PROBLEM DEFINITION AND RELATED WORK	 7
2.1 Cell Layout Styles	7
2.1.1 Gate Matrix Layout Style	8
2.1.2 Line of Diffusion Layout Style	8
2.1.3 Flexible Layout Styles	11
2.2 The Synthesis Process	11
2.2.1 Transistor Level Design	12
2.2.2 Transistor Level Placement	12
2.2.3 Routing of Internal Nets	13
2.2.4 Detailed Layout Synthesis	14
2.3 Algorithms for Synthesis of Arbitrary Circuits	14
2.3.1 Description of Sample Methods	14

2.3.2 Comparison of Transistor Placement Techniques	16
2.4 Synthesis of CMOS Complex Gates	17
2.5 Knowledge-Based Synthesis	19
2.5.1 Topologizer	19
2.5.2 TALIB	20
2.6 Overview of Symbolic Layout Compaction	21
2.7 Summary	22
Chapter 3: THE PICASSO APPROACH	23
3.1 General Overview of the System	23
3.2 Synthesis Method	24
3.2.1 Transistor Level Synthesis	25
3.2.2 Topological Expansion	27
3.2.3 Detailed Layout Synthesis	27
3.3 Data Representation	29
3.3.1 Symbolic Representation of Layout Objects	29
3.3.2 Representation of Connectivity	30
3.4 Implementation Paradigms	32
3.5 Summary	35
Chapter 4: SYNTHESIS OF COMPLEX GATES	36
4.1 Definition of the Graph Model	37
4.2 General Algorithm	40
4.2.1 Graph Traversal Example	40
4.2.2 Determination of Sub-graph Types	42
4.2.3 Graph Traversal Strategy	43
4.3 Advanced Features	46
4.3.1 Grouping of Sub-graphs	46

4.3.2 Splitting of Gate Signals	48
4.3.3 Analysis of Input Signals	51
4.3.4 Interconnect Considerations	52
4.4 Summary	54
Chapter 5: TOPOLOGICAL CELL EXPANSION	55
5.1 Cluster Placement	55
5.1.1 Representation of Cluster Morphology	56
5.1.2 The Connectivity Graph	57
5.1.3 Affinity Between Clusters	58
5.1.4 Placement Algorithm	60
5.2 Routing of Internal Nets	63
5.2.1 Routing Regions	63
5.2.2 Net Types	63
5.2.3 Assignment of Nets to Regions	64
5.2.4 Ordering Nets within Regions	65
5.2.5 Topological Compaction	65
5.2.6 Creation of Routing Grid	66
5.2.7 Routing Example	66
5.3 Coupling Between Placement and Routing	69
5.4 Symbolic Layout Refinement	70
5.4.1 Implementation of I/O Ports	70
5.4.2 Horizontal Position of Transistors	71
5.4.3 Orientation for Substrate Contacts	71
5.5 Summary	71
Chapter 6: FLEXIBLE BOUNDARIES FOR LAYOUT SYNTHESIS	72
6.1 The Concept of Flexible Boundaries	73

6.2 Implementation of Flexible Boundaries	75
6.2.1 Data Representation	75
6.2.2 Updating Boundaries	77
6.3 Determining the Position of Objects	80
6.4 Determining the Shape of MOS Transistors	82
6.5 Determining the Shape of Wires	84
6.5.1 General Wire Generation Algorithm	84
6.5.2 Generation of Wires Constrained on Both Sides	85
6.6 Summary	89
Chapter 7: CMOS LAYOUT SYNTHESIS USING FLEXIBLE BOUN- DARIES	90
7.1 Boundary Definitions for CMOS Layout	90
7.1.1 Boundaries for the Polysilicon Layer	90
7.1.2 Boundaries for the Diffusion Layer	92
7.1.3 Boundaries for the Metal Layer	94
7.1.4 Other Boundaries	96
7.2 Outline of the Cell Layout Algorithm	97
7.2.1 Formation of Symbolic Layout Slices	97
7.2.2 Generation of Layout Geometries	97
7.2.3 Representation of Constraints	98
7.2.4 Summary of Layout Synthesis Algorithm	98
7.3 Creating Lines of Diffusion	99
7.3.1 Generation of Wide Transistors	100
7.3.2 Generation of Weak Transistors	102
7.3.3 Generation of Diffusion Contacts	103
7.3.4 Generation of Diffusion Bridges	104
7.4 Generating Geometries for Polysilicon Objects	105

7.4.1 Generation of Polysilicon Contacts	105
7.4.2 Generation of Polysilicon Wires and Ports	105
7.5 Generating Geometries for Metal Wires	107
7.6 Summary	107
Chapter 8: EXPERIMENTAL RESULTS	108
8.1 JK Flip-Flop Example	108
8.1.1 Transistor Level Design	109
8.1.2 Cluster Placement	111
8.1.3 Routing of Internal Nets	115
8.1.4 Layout Synthesis	118
8.2 Alternative Cell Structure	120
8.3 Logic Synthesis Example	123
8.4 Impact of Qualitative Process Description	128
8.5 Complexity Issues	130
Chapter 9: CONCLUSIONS AND FUTURE WORK	132
9.1 Contributions	132
9.2 System Limitations	134
9.3 Future Work	135
Appendix A: GLOSSARY OF TERMS	136
Appendix B: GATE PATTERNS FOR CLUSTER FORMATION	138
BIBLIOGRAPHY	140

LIST OF FIGURES

Number	Title	Page
1.1	ASIC design flow with cell synthesis	4
2.1	Gate matrix layout style	9
2.2	Variations of the line of diffusion layout style	10
2.3	Graph model proposed by [UevC]	18
2.4	Physical constraint representation using graph framework	22
3.1	Picasso block diagram	25
3.2	Cell structure used by Picasso	28
3.3	Object-based symbolic layout primitives	30
3.4	Representation of connectivity via pointers	31
3.5	Representation of connectivity on the metal layer	33
4.1	Mapping between graph models used in [UevC] and Picasso	37
4.2	Graph model example	39
4.3	Example of general synthesis method for complex gates	41
4.4	Summary of sub-graph types	42
4.5	Graph traversal procedures	44
4.6	Optimal solution found by grouping sub-graphs	47
4.7	Use of gate splitting to achieve optimal chaining	49
4.8	Use of gate splitting for chaining with adjacent cells	50
4.9	Use of repeated input signals in graph traversal	51
4.10	Assigning graph nodes to diffusion signals	53
5.1	Tile representation for clusters	56
5.2	Computation of morphology-based affinity	59
5.3	Placement selection for new cluster	62

5.4	Simple routing example	67
6.1	Flexible boundary concept	74
6.2	Flexible boundary representation	76
6.3	Adding segments to a boundary definition	78
6.4	Determination of the nature of segments to be added	79
6.5	Determining the position of an object of fixed shape	81
6.6	Determining the shape of MOS transistors	83
6.7	Sample filter function for the generation of wires	84
6.8	Patterns used for boundary filtering	86
6.9	Wire segments corresponding to boundary segments	87
6.10	Generation of wires constrained on both sides	88
7.1	Boundary definitions for the polysilicon layer	91
7.2	Boundary definitions for the diffusion layer	93
7.3	Boundary definitions for the metal layer	95
7.4	Dual purpose of flexible boundaries	99
7.5	Generation of wide transistors	101
7.6	Weak transistors supported by Picasso	102
7.7	Diffusion contact structures	103
7.8	Generation of diffusion bridges	104
7.9	Generation of polysilicon contacts	106
8.1	JK flip-flop logic schematic	109
8.2	Transistor level structures generated for JK flip-flop	110
8.3	Data structures used for cluster placement operation	112
8.4	Trace of placement algorithm for JK flip-flop	113
8.5	Detailed transistor placement for JK flip-flop	114
8.6	Structures provided as input to the routing algorithm	116

8.7	Detailed symbolic representation for JK flip-flop	117
8.8	Partial layouts after first few slices for JK flip-flop	118
8.9	Complete layout generated for JK flip-flop example	119
8.10	Symbolic layout for JK flip-flop under new cell structure	121
8.11	Detailed layout for JK flip-flop under new cell structure	122
8.12	Full adder logic description	123
8.13	First outline generated for full adder	124
8.14	Second outline generated for full adder	125
8.15	Symbolic layout generated for full adder example	126
8.16	Detailed layout generated for full adder example	127
8.17	Detailed layout for full adder with Manhattan geometries	129
B.1	Summary of gate patterns for complex gates	138
B.2	Summary of gate patterns for pre-defined structures	139

LIST OF APPENDICES

Appendix	Title	Page
A	Glossary of Terms	136
B	Gate Patterns for Cluster Formation	138

**To
My Wife
Nirmala**

CHAPTER 1

INTRODUCTION

1.1. OBJECTIVES

The debate over the relative merits of automatic leaf cell generators versus carefully hand-crafted layouts is at its height. With recent advances in Very Large Scale Integration (VLSI) technologies and Computer-Aided Design (CAD) systems, the field of leaf cell generation which had been confined to the academic scene, is now taking giant strides towards general acceptance in the industry. The proponents of general purpose leaf cell generators speak of the immense potential of this new technology and contend that it is in keeping with the general evolution of CAD for VLSI. A strong argument often put forward in favor of layout synthesis tools is that they provide a direct path to process independent designs. This greatly facilitates the upgrade of existing designs to new technologies as well as second sourcing. In addition, the ability to customize individual cells to their environment introduces the opportunity for a number of optimizations not possible in library-based systems. Hence, cell synthesis should be seen as an enabling technology, which is a vital part of an enhanced CAD methodology.

The opposing view is generally based on more subjective considerations. First, the increased complexity of the CAD system poses a certain threat to the integrity of the resulting designs. Since a thorough characterization of individual cells is excluded, design verification must now rely on more sophisticated process and circuit simulation tools and models. Secondly, one could argue that a truly general cell compiler is itself very difficult to test and may generate unpredictable results.

The objective of this thesis is therefore to demonstrate the feasibility of an automatic leaf cell generator capable of emulating manual designs, both in terms of area efficiency and layout style. It describes a number of algorithms, data structures and ideas that have already contributed to the state-of-the-art in cell synthesis. The implementation of a complete prototype is also presented along with extensive experimental results.

1.2. MOTIVATION

In order to gain a full appreciation for the usefulness of leaf cell synthesis tools, one must look at their impact on the design flow for intended applications.

1.2.1. Impact on ASIC Design Flow

Application Specific Integrated Circuits (ASIC), as the name implies, are circuits designed with a particular use in mind, as opposed to their general purpose counterparts. This often implies relatively low volume and short product life time, and hence, little opportunity for recuperating the design cost. For that reason, a cost/performance trade-off is often achieved by using design methodologies such as Standard Cells and Gate Arrays. The standard cells methodology reduces the IC design task to a module placement and interconnect problem. Gate arrays impose a further restriction on the possible module placement sites, which allows the repeated use of regular layout templates. In both cases, module selection is performed from a fixed library of pre-designed cells and/or special purpose generators.

Leaf cell synthesis opens the door to a new CAD methodology for cell based ICs, as illustrated in Figure 1.1. Indeed, general purpose leaf cell generators allow designs to be captured in a generic form, using a library of logic primitives, as opposed to physical modules. These primitives, which only represent the intended behavior, are at

the root of generic design capture tools. A design hierarchy can be introduced by grouping logic primitives into leaf cells for layout generation.

The resulting design flow can be summarized as follows. From the logic net list and estimates on the area required for each leaf cell, an approximate floor plan and cell placement are obtained. This approximate layout is then used to compile a detailed environment description for each cell, comprised of both electrical and physical parameters. Information regarding capacitive loading on each output is useful in optimizing transistor sizes on selected critical paths, whereas the physical environment description can help determine a preferred ordering and layer assignment of input/output (I/O) ports for each cell. At this point, based on the logic net list, transistor sizes and physical constraints, detailed layout geometries can be generated for each cell, and used to produce a layout for the complete circuit. If necessary, this detailed layout can be fed back as a new estimate, and the design iterated upon until specific requirements are met. The opportunity for post-layout leaf cell optimization is the key benefit provided by this new design methodology.

1.2.2. Application to Logic Synthesis

Another target application is logic synthesis, which consists of transforming arbitrary boolean equations into a layout. This process involves logic manipulation and minimization, followed by a mapping into the target technology and layout style. Regular structures such as Programmable Logic Array (PLA) and Weinberger Array are often used for layout. In general, PLAs are limited to sums of products whereas Weinberger arrays can accommodate forms of logic of arbitrary depth. The area efficiency and speed of the design both decrease quite rapidly with increasing depth.

Alternatively, technology mapping can be based on a library of logic functions such as and-or-invert (AOI) cells. For practicality reasons, the depth of logic functions

Figure 1.1. ASIC design flow with cell synthesis.

implemented at the cell level is normally limited to two. Clearly, the ability to synthesize arbitrary AOI functions on demand is a significant asset. The cell selection needs no longer be limited to a fixed number of pre-designed cells. Also, the depth of logic functions, implemented by leaf cells, can now be increased, making the system more flexible. Furthermore, all the optimization arguments put forward in the above discussion about ASICs hold true in the context of logic synthesis as well.

1.2.3. Impact on Design Portability

Finally, leaf cell synthesis tools provide increased flexibility in terms of producing "process independent" designs. Design portability refers to the ability to transport designs between processes, which may have different layer definitions, as well as distinct physical and electrical design rules. A number of factors contribute to the fundamental difficulty of achieving process independence. The most important factor resides in the fact that design rules associated with different processes will not only differ quantitatively, but may also exhibit certain fundamental qualitative differences. For example, the number and the nature of the layers present may vary, as well as the definition of the interaction between them. Furthermore, certain geometric structures may be allowed in one process and constitute violations in another.

Attention must also be given to electrical considerations, which may preclude simple design scaling, due to its drastic effect on capacitive loading. Indeed, processes having thinner gate oxides also have increased loading conditions. Consequently, optimal transistor sizes may vary considerably when porting designs between processes, requiring a change in cell topology. Therefore, the ability to consider device sizes in the creation of a cell's topology is a very important feature, and introduces the opportunity for a better exploitation of new technologies.

1.3. OUTLINE OF THE THESIS

Chapter 2 is dedicated to describing the cell synthesis problem, and reviews important related work from the literature. In particular, different cell layout styles are compared in terms of their impact on the synthesis and the quality of the resulting layouts. Chapter 3 describes the approach used in the Picasso cell compiler. It discusses the aim of the system and the key features that distinguish it from existing systems. It also addresses issues of circuit representation, implementation paradigm and synthesis method. Chapter 4 discusses an important transistor level design problem: the automatic synthesis of CMOS complex gates. Methods are described for mapping logic level descriptions into a transistor net list which is optimized for the target layout style. The purpose of Chapter 5 is to describe methods for mapping a transistor level net list into a symbolic layout. It addresses the issues of topological ordering of layout objects in preparation for layout synthesis. Chapter 6 discusses flexible boundaries, a data structure used by Picasso for representing physical constraints between layout objects. Chapter 7 illustrates the use of flexible boundaries in CMOS layout synthesis. It describes a complete algorithm for the generation of layout mask geometries based on a symbolic representation. Finally, Chapter 8 presents experimental results, followed by conclusions, contributions and future work, in Chapter 9.

CHAPTER 2

PROBLEM DEFINITION AND RELATED WORK

The leaf cell synthesis problem consists of automatically generating an efficient layout based on a higher level description which can either take the form of a transistor netlist or a logic schematic, depending on the scope of the system and the target design style. For example, static circuits can be adequately represented using a combination of logic gates and transmission gates, whereas other design styles such as dynamic circuits and pass transistor circuits may require a detailed transistor level representation. Strictly speaking, there is no restriction on the number of transistors included in each leaf cell. However, for the purpose of this discussion we will assume that problem sizes are limited to fewer than a hundred transistors and that larger structures can be built hierarchically.

This chapter reviews the fundamental issues in cell synthesis such as layout styles and general methods. It describes the various steps of the leaf cell synthesis process, the major difficulties associated with each one, as well as a number of representative solutions from existing systems.

2.1. CELL LAYOUT STYLES

A layout style is a design framework that imposes some physical restrictions on the synthesis process. Its purpose is to simplify the synthesis problem itself as well as facilitate interfacing leaf cells to their external environment. A layout style typically specifies a cell's pitch or aspect ratio, provides a general template for transistor level placement and routing, and defines a preferred I/O implementation strategy.

Cell layout styles can be divided into two groups: (i) architectures geared towards implementation of large combinatorial circuits such as PLAs and Weinberger Arrays, and (ii) general purpose architectures which can be used to implement arbitrary circuits. This section focuses on the latter, and discusses the relative merits of the two most popular approaches. These are the gate matrix and line-of-diffusion layout styles.

2.1.1. Gate Matrix Layout Style

In the gate matrix layout style (see Figure 2.1), all transistors with a common gate connection are located on the same vertical polysilicon line [LoLa], [KKLL]. Hence, p-type and n-type devices form two-dimensional arrays at the top and bottom of the cell, respectively. Interconnections between the devices are implemented in the routing area at the center of the cell.

A number of methods for automatic cell synthesis based on the gate matrix layout style have been presented in the literature [Wing], [WiHW], [Li]. Although some of these methods use very sophisticated algorithms and data structures, they are all confronted with the inherent limitations of an over-constrained layout style. Key among those limitations, is the fact that different transistors must have the same gate signal in order to be allowed to share a column. Furthermore, gate matrices cannot gracefully accommodate a variety of transistor sizes. Hence, for best area efficiency, the transistors used in the layout should all be of the same size.

2.1.2. Line-of-Diffusion Layout Style

The line-of-diffusion layout style, as the name implies, consists of two parallel lines of diffusion for p-channel and n-channel devices, respectively. Polysilicon gates run perpendicular to the diffusion lines, whereas metal connections run parallel to them.

Figure 2.1. Gate matrix layout style.

Two variants exist for this layout style. The first one, shown in Figure 2.2 (a), is analogous to a one-dimensional gate matrix, since it does not allow an overlap of routing and active areas. This layout style is very popular for synthesis since it creates a clear separation between the placement and routing problems. It is used in a number of systems including those described in [BaAl], [NaBR], [WiPF] and [Hill].

The second variant is obtained by allowing an almost complete overlap of routing and active areas, leaving just sufficient spacing between p-type and n-type devices as specified by process design rules. This layout style which is commonly adhered to by human designers for standard cell libraries, yields uniform pitch cells. This constitutes an important feature when automatic placement and routing tools are to be used. This

(a) One-dimensional gate matrix

(b) Lines of diffusion with overlap of routing region

Figure 2.2. Variations of the line-of-diffusion layout style.

layout style also promotes electrical optimization of the design by allowing transistors to be sized individually. The cost of handling devices of different sizes is minimized by the fact that metal connections can be routed over the active cell area. Hence, area left vacant by transistors that are smaller than the maximum size allowed, can easily be recuperated by implementing contact windows or vias. Figure 2.2 (b) illustrates this cell architecture.

2.1.3. Flexible Layout Styles

An alternative to the above mentioned techniques, is the absence of a uniform layout style. Individual constraints can be applied to each cell. This promotes a top-down design strategy in which the aspect ratio for each cell is defined before a layout can be generated. Such layout styles may generate denser layouts due to their potential for eliminating wasted area. On the other hand, fewer restrictions result in a far more complex synthesis problem. This increase in problem complexity has led to the proposal of knowledge-based synthesis techniques [KoWe], [KiMS], as discussed in a later section.

2.2. THE SYNTHESIS PROCESS

The cell synthesis process can be defined as the joint solution of four sub-problems: (i) transistor level design, (ii) device placement, (iii) routing and, (iv) generation of layout geometries. The nature of each sub-problem and associated objective function are described in this section.

2.2.1. Transistor Level Design

One of the key issues in cell synthesis that is very seldom addressed by existing systems is the mapping of logic descriptions into efficient transistor level net lists. This constitutes a one-to-many mapping which has a direct incidence on the efficiency of compiled layouts for two main reasons. The first is the fact that, of all the possible transistor structures corresponding to some logic description, only a small subset will contain an optimal number of transistors. Furthermore, transistor net lists which are optimal in terms of the number of transistors may not lend themselves equally well to efficient layouts.

Efficient transistor level designs may only result from a reasonable trade-off between potentially conflicting objectives, as well as a thorough analysis of the target cell structure. For example, it is desirable in most layout styles to create maximal chains of transistors that can share common diffusion regions. Although merging transistors through common diffusions has a general tendency to reduce the amount of metal connections inside the cell, interconnect considerations may have to play a more explicit role. Indeed, when a hard limit is imposed on the number of metal tracks available within the active cell area, sub-optimal transistor chains may have to be selected in order to meet the design requirements. Similarly, the benefit of optimal transistor chains may have to be weighted against the vertical alignment of multiple transistor gates. Thus, it is crucial to consider the device placement template, specified by the design style, when generating transistor level structures.

2.2.2. Transistor Level Placement

Following the generation of a transistor level net list, it is necessary to determine the relative placement of each transistor according to the selected layout style. For example, using gate matrices, the objective of the placement phase is to align

transistors that share a common gate signal into a single column. The principle objective of the line-of-diffusion layout style is to maximize common diffusion connections. Again, minimizing the total amount of metal interconnect may be an important consideration. The overlapping objective functions between transistor level design and transistor placement suggests that these two task may be best achieved simultaneously.

Transistor placement may also be subjected to physical constraints such as the ordering of I/O ports, minimizing the span of critical nets and accommodating a variety of transistor sizes. Furthermore, provisions may have to be made for substrate contacts and feed-through nets.

2.2.3. Routing of Internal Nets

The next step consists of routing internal nets necessary to complete the cell's functionality. Again, the nature of the problem depends on the design style selected. In layout styles which do not allow an overlap of routing and active areas, the internal net routing problem is reduced to channel routing. All the points to be connected are located at the top and bottom of the routing area at the center of the cell. However, for less restrictive layout styles that do allow routing over the active areas, the problem is considerably more complex, since it involves routing wires between points that can be anywhere within the routing area, in addition to the top and bottom. Such cases require that the construction of underlying devices geometries be performed in parallel with internal routing.

Another important consideration may affect the objective of the routing phase. The routing region can either be rigid, or may be allowed to expand to arbitrary dimensions. For fixed pitch cells, the objective is to complete the routing in the available area, whereas for variable pitch cells the goal is to minimize the increase in cell area necessary for routing.

2.2.4. Detailed Layout Synthesis

Detailed layout synthesis is the last step of the synthesis process, which consists of generating mask data for the circuit. Most cell synthesis systems produce a symbolic layout as output and rely on traditional fleshing and compaction methods for generating the layout. Key issues in symbolic layout compaction are addressed in Section 2.6.

2.3. ALGORITHMS FOR SYNTHESIS OF ARBITRARY CIRCUITS

This section presents a sample of the most important systems found in the literature, for the generation of arbitrary circuits. The study of more specific problems, such as complex gates, is the subject of the next section. The vast majority of existing systems, operate from a transistor level input description and produce a symbolic layout as output. The layout style selected almost invariably creates a clear distinction between active and routing regions. The routing problem is therefore treated as a channel routing problem, in most cases. Hence, this section attempts to highlight the differences between these systems in terms of their transistor level placement techniques.

2.3.1. Description of Sample Methods

Optimal chaining of transistors is the objective of the algorithm presented in [WiPF]. Transistor placement is based on the creation of "chains of transistor pairs", where transistors that are vertically aligned must have a common gate signal. Maximal chains are formed by abutting transistors with common diffusion ports and are assigned a cost reflecting their merit. A sub-set of the chains is selected to realize the circuit using an enumerative algorithm. Following chain selection, a relative placement is obtained using an exhaustive search. The routing problem that remains is analogous

to the channel routing problem, since all intracell routing is implemented in between the two transistor chains.

The system presented in [Hill], can also handle arbitrary circuits. Transistor placement is based on heuristic pairing of transistors, requiring two transistors in a pair to have a common gate signal. A min-cut placement technique is then applied to the pairs which attempts to minimize the amount of internal connections required. The placement is refined by "flipping" transistors to increase the number of common diffusion connections. The routing algorithm attempts to route connections over the active areas whenever feasible and terminates with channel routing of remaining connections at the center of the cell.

SOLO [BaAl] is a program for generating a symbolic layout, given a transistor level schematic and a set of boundary constraints. The strength of the system resides in its ability to handle arbitrary transistor sizes as well as an unequal number of n-type and p-type devices. SOLO first partitions the circuit into series-parallel combinations of similarly sized, identically typed transistors. Each sub-circuit is then mapped onto a one-dimensional gate matrix module. A general placement technique is used to establish the final position for each module within the cell.

An algorithm described in [SSKT] produces cell layouts analogous to standard cells. The device placement method attempts to minimize the number of diffusion separations, straighten gate wires between p-type and n-type transistors and minimize the total wiring length for internal nets. The routing algorithm recognizes five distinct routing channels. One channel is defined over each row of transistors as well as between them. In addition, two more channels are defined below the lower row of transistors and above the upper row, respectively. These two additional channels are only used as a last resort. The routing process consists of three sub-processes: (a) channel selection, (b) detailed routing within each transistor row, and (c) detailed

routing in remaining three channels.

Stauffer and Nair [StNa] have recently reported on a relaxation approach to the transistor placement problem. They have applied simulated annealing [KiGV] and iterative improvement techniques [Lin] to the one-dimensional device placement problem. The method is based on an objective function which includes a measure for routability as well as the number of diffusion breaks.

2.3.2. Comparison of Transistor Placement Techniques

The above mentioned methods differ significantly in terms of their placement strategy. The method presented in [WiPF] attempts to minimize the number of diffusion separations while vertically aligning transistors with common gate signals. Internal routing minimization is given a secondary consideration. In the same line of thought, [Hill] considers interconnect first by using a min-cut placement technique on the transistor net list. An attempt to minimize diffusion gaps comes second, through the use of "transistor flipping".

On the other hand, the "Diffusion Island Couple" algorithm [SSKT], attempts to give equal weight to each criterion, by performing cell optimization in two steps. First, minimization of the number of diffusion gaps is emphasized, to obtain an initial placement which is then refined by considering interconnect. Similarly, Stauffer and Nair [StNa] allow the objective function used in their relaxation technique to be customized according to the desired layout style. This feature makes it possible for the user to fine tune the weighting attributed to routability.

2.4. SYNTHESIS OF CMOS COMPLEX GATES

Complex gates are combinatorial circuits implemented as series-parallel combinations of transistors. Such circuits lend themselves naturally to efficient layouts in the line-of-diffusion layout style, and hence, constitute an important sub-problem. Efficient algorithms for the automatic synthesis of complex gates are important, in particular, in logic synthesis, as discussed in the introduction. However, complex gates also find an application in the synthesis of more general circuits. Indeed, grouping into complex gates, individual AND and OR gates, may result in a significant reduction in the number of transistors required for their realization.

The search for an optimal realization for CMOS complex gates was pioneered by Uehara and vanCleemput [UevC] who proposed a graph-theoretical framework for layout synthesis, shown in Figure 2.3. The graph is separated into a p-side and an n-side representing p-MOS and n-MOS portions of the circuit respectively. Every source/drain potential is represented as a vertex and labeled with the corresponding signal name. Every transistor is represented by an edge connecting the source and drain vertices and is labelled with the gate signal name. If two edges are adjacent in the graph, then it is possible to place the corresponding transistors side by side, in the layout, and to connect them via a common diffusion.

The key point is that if there exists an Euler path, (i.e. a sequence of edges that traverses all the edges of the graph exactly once), then all the transistors can be chained into a single diffusion area. If there is no such path, then the graph must be decomposed into several sub-graphs for which Euler paths exist, consequently introducing gaps in the line of diffusion. Also, if the ordering of gate signals is the same for both p-side and n-side graphs, then complementary transistor pairs can be vertically aligned in the layout, allowing trivial connection of their gate terminals with polysilicon wires alone. Such an ordering is referred to as a dual-Euler path.

Figure 2.3. Graph model proposed by [UevC].

In addition to a graph model, Uehara and vanCleave also proposed a set of heuristics for cell synthesis based on graph reductions [UevC]. Their algorithm is exhaustive in nature and has been shown not to be optimal [MaHa]. Furthermore, it imposes the restriction that complementary transistor pairs be vertically aligned in the layout.

The work of Uehara and vanCleave was later generalized by Maziasz and Hayes [MaHa], who proposed an optimal non-exhaustive method based on the same cell model, and which exhibits linear time complexity in the number of transistors used. However, they do not address the problem of reordering transistors in the graph model, which constitutes a significant limitation.

The efficient layout of complex gates was also studied by Nair, Bruss and Reif [NaBR]. They have proposed linear time algorithms for finding both the optimal graph and optimal layout for a given circuit when such an optimal solution exists.

Their algorithms also impose the restriction that complementary transistor pairs be vertically aligned. Hence when an optimal solution does not exist, gaps are introduced in the diffusion lines.

Finally, Mailhot and DeMicheli [MaDe] have proposed an alternate approach, which exploits the fact that Boolean equations can be represented in Backus-Naur form (BNF). The BNF is a tree-like representation, in which leaf nodes represent literals and non-leaf nodes correspond to either a sum or a product of two boolean factored forms. Such a representation allows series-parallel transistor structures to be build in linear time, using a set of simple composition rules. Hence, their method does consider the important problem of reordering transistors in the circuit description.

2.5. KNOWLEDGE-BASED SYNTHESIS

This section discusses the application of rule based expert systems to cell synthesis. Both systems discussed herein use a flexible layout style.

2.5.1. Topologizer

Topologizer is a rule-based automatic synthesis system for leaf cell design [KoWe]. Input to Topologizer consists of a transistor connection list and an environment description which contains the desired cell aspect ratio and external signal placement. Following a random initial transistor placement, a 20-rule expert system attempts to find a good placement, by representing each transistor as a tile. Each tile has three sections corresponding to the source, gate and drain of the transistor, which makes the placement process analogous to the game of dominos. The objective is to maximize the number of adjacent domino nodes that match.

After completing the placement, Topologizer invokes the routing expert. Again, a correct but wasteful initial solution is found and then improved upon iteratively. The result is a symbolic description of the desired cell to be passed to a symbolic layout system for layout compaction.

The techniques used by Topologizer are guaranteed to yield a solution, the quality of which depends on that of the placement and routing expert systems. However, the disjoint nature of these two operations represents a fundamental limitation. Furthermore, placement based on individual transistors may not be practical for very large cells with fixed pitch constraints.

2.5.2. TALIB

TALIB is a knowledge-based expert system for cell layout, tailored for a single metal layer nMOS process and a Manhattan geometry [KiMS]. It capitalizes on the fact that typical circuits are built out of a number of commonly occurring substructures. Domain specific knowledge is used to recognize certain patterns and manipulate them to determine the relative placement of individual transistors. The user specifies transistor interconnects as well as a number of layout constraints (terminal position, feed-through nets, routing layer and design rule information) and TALIB produces final layout masks by manipulating a certain number of "layout models", which are pre-defined geometric structures corresponding to circuit elements. Performance of the system has been found to be dependent on the number of transistors in the input description as well as the boundary conditions specified by the user. The main drawback to the TALIB system is the use of pre-defined "layout models". Such models severely limit the flexibility of a layout system.

2.6. OVERVIEW OF SYMBOLIC LAYOUT COMPACTION

Most of the systems discussed to this point, treat the cell synthesis problem as topological problem without addressing the issue of geometric mapping. For the generation of detailed layout geometries, many systems rely on symbolic layout compaction. Several different approaches have been proposed for solving the compaction problem [ChKS], [Dunl], [Most], [WMND], [Cho], [BoWe]. This section focuses on the constraint-graph method which has become clearly predominant.

Constraints-based symbolic layout compaction consists of two main steps: (i) constraint-graph generation, to indicate relative positions, minimum separations as well as connectivity among layout objects, and (ii) constraints solution to determine absolute positions. Figure 2.4 summarizes the graph representation for physical constraints between layout objects. Figure 2.4 (a) illustrates the graph-theoretical elements used to represent a separation constraint between rectangles P-1 and P-2. Minimum feature sizes can be expressed in a similar fashion. On the other hand, overlap rules and electrical connectivity between two rectangles can be expressed using negative constraints, as shown in Figure 2.4 (b). Recent publications have proposed improvements to the basic method. The objective of "supercompaction" [WMND], for example, is to apply geometric reorganizations to the symbolic representation in order to break critical separation constraints, and to introduce "jogs" in wires to break connectivity constraints. Other extensions to the constraint-graph method, to accommodate 45 degree bends in wires, have also been presented [Sun] [MaSH].

In addition to explicit constraints, a data structure called "contour" can be used to represent minimum spacing constraints on a single layer. This concept was first introduced in the context of optimal river routing [Ullm] [Hsu], and was later applied to channel compaction as well [XiKu], [Deut], [RPVN]. Contours have also been used to enhance the functionality of constraint-based compaction, in a method called zone refining [SSVS].

(a) Representation of separation constraint

(b) Representation of overlap constraint

Figure 2.4. Physical constraints representation using graph framework.

2.7. SUMMARY

The leaf cell synthesis problem encompasses a wide variety of issues which include transistor level design, device level placement, routing and symbolic layout compaction. The selection of a layout style has a direct impact on the synthesis process. In particular, a number of cell synthesis algorithms have been proposed, which are based on the gate matrix and the line of diffusion layout styles. The study of CMOS complex gates is of particular interest due to their direct applicability to logic synthesis.

CHAPTER 3

THE PICASSO APPROACH

This chapter provides a general introduction to the Picasso CMOS leaf cell generator. The emphasis is on the high level behavior of the system rather than the subtle details of the individual synthesis steps. A more in depth description of the algorithms and data structures used is found in subsequent chapters. The synthesis method used in Picasso builds upon certain ideas presented in existing synthesis systems, combined with a thorough analysis of the manual layout process. As a result, it is able to consider all the important cell design metrics, while emulating hand designs both in terms of compactness and layout style. Although Picasso is presented here as a synthesis tool for static CMOS circuits, many of its building blocks are generic in nature and could be modified to generate dynamic circuits as well, and could even be applied to other technologies.

3.1. GENERAL OVERVIEW OF THE SYSTEM

The purpose of the Picasso system is to map a gate-level description into a detailed physical representation. The input description is a net list comprised of logic gates and transmission gates, to allow the specification of both combinatorial and sequential circuits. Hence, the synthesis process includes the mapping gate-level descriptions into transistor structures.

Layouts produced by Picasso are based on the line-of-diffusion layout style. In order to facilitate the use of synthesized cells by higher level tools, a number of common characteristics may be specified by the user, for a number of different cells. Such common characteristics include, but are not limited to, the absolute position and width of Vdd and Vss busses as well as the upper and lower boundaries of each active

region. In addition, it is possible for the user to specify a discrete grid for the alignment of I/O points.

Picasso accepts as part of its input, a process description which contains quantitative information concerning the numerical values of physical dimensions, spacings and overlaps, as well as a qualitative description of the nature of the various layers. For example, the qualitative section is used to describe the geometries allowed on each layer, such as forty-five degree bends in transistor gates. It may also include information regarding the presence or absence of silicidation on the diffusion and polysilicon layers, that will help determine the implementation of selected layout features.

Finally, Picasso provides the necessary hooks for the specification of individual transistor sizes, either by the user or through the use of an electrical optimization tool. Although the ability to handle arbitrary transistor sizes is not new, Picasso's synthesis method is such as to take full advantage of the space left vacant by small transistors and minimize the cost of large transistors by appropriately bending layout features. Even weak transistors, for which $W < L$, are handled gracefully.

3.2. SYNTHESIS METHOD

Picasso partitions the cell synthesis problem into three sub-tasks, each associated with a software module, as shown in Figure 3.1. These are: (i) transistor level design, (ii) topological expansion, which encompasses device placement, as well as routing of internal nets, and finally (iii) generation of layout geometries. In addition, a fourth module, the environment compiler, serves to combine the user supplied process description file and a list of common constraints, into a collection of static parameters, used by the other three modules, throughout the synthesis process.

Figure 3.1. Picasso block diagram.

3.2.1. Transistor Level Synthesis

Transistor level synthesis consists of generating a transistor level net list corresponding to the input logic description. The objective is to minimize the number of transistors used while ensuring that the resulting net list is amenable to an efficient realization in the target layout style. Given that Picasso uses the line-of-diffusion layout style, there is much to be gained by generating transistor net lists with placement in mind, and eliminate solutions that appear optimal at the transistor level but yield poor layouts under the given set of constraints. Therefore, the layout strategy is to map the logic descriptions onto a transistor net list represented as two lines of diffusion corresponding to p-type and n-type transistor chains, respectively.

The direct translation of logic primitives into corresponding transistor structures is both near-sighted and wasteful. Indeed, it is necessary to first recognize logic structures, such as AND-OR-INVERT gates, that, when considered as a whole, yield far more efficient solutions than the combination of the solutions obtained for individual gates. Hence, the first step in the design flow is a process called clustering. It consists of grouping the logic primitives of the cell description, by assigning them to clusters, according to their amenability to efficient layouts in the line-of-diffusion layout style. The purpose of the clustering phase is twofold. First, it identifies complex gates and other structures for which well defined procedures exist to generate efficient transistor structures. Clustering also serves to simplify the transistor placement task by introducing a placement hierarchy.

The cluster formation process is carried out by traversing the input logic net list. A search is conducted beginning at each primary output and terminating at inputs. Each gate that is encountered, that has not yet been assigned to a cluster, is used as a seed for the formation of a new cluster. New clusters are formed by comparing the local net list with pre-defined gate patterns. The next step consists of generating a transistor net list corresponding to each cluster. In some cases, such as for inverters or transmission gates this is a very straightforward one-to-one mapping. In other situations, however it constitutes a one-to-many mapping where a number of constraints have to be considered. A graph-theoretical framework has been developed, for such complex cases, and is discussed in detail in Chapter 4. Both the direct translation of simple gates and the graph-theoretical mapping of complex gates produce transistor net lists represented as two lines of diffusion. Hence, the relative ordering of transistors within clusters is defined at this point. Appendix B summarizes the patterns used in the current prototype for cluster formation.

3.2.2. Topological Expansion

Topological expansion consists of generating a symbolic layout representation for the cell, in preparation for detailed layout synthesis. It encompasses the tasks of transistor level placement and routing of internal nets, described in Chapter 2. The merger between the placement and routing phases is a consequence of the layout style chosen, which allows a complete overlap of routing and active areas. The construction of a symbolic representation is guided by a layout template which partitions the cell area into three regions, as shown in Figure 3.2. The p-region and n-region are used to implement chains of p-type and n-type devices, respectively, and the center-region is used exclusively for routing internal connections. As mentioned above, this layout template may be customized by the user. It contains a set of parameters which determine the vertical position of Vdd and Vss busses as well as the upper and lower limits of each region. The position of metal tracks within the cell may be specified explicitly or in the form of a "vertical range" within which individual wires may be positioned for each metal track.

During the topological expansion process, symbolic objects such as transistors and contact windows are assigned a vertical position and become like the wooden balls of an abacus, which are constrained in one direction but free to move from left to right. Electrical nets are assigned to physical layers and a two-dimensional topological ordering is determined among the various objects. Chapter 5 addresses the topological expansion problem.

3.2.3. Detailed Layout Synthesis

Layout synthesis is the final step of the cell generation process, where the symbolic layout representation is mapped onto mask geometries. The approach taken by Picasso consists of first decomposing the symbolic representation into "slices",

Figure 3.2. Cell structure used by Picasso.

corresponding to vertical virtual grid lines. The algorithm operates from left to right by processing each slice in turn. Following the generation of layout geometries for all the objects and wires present in the slice definition, physical constraints are computed in order to restrict the geometries generated for the following slice. The exact shape and position of layout objects is determined at this point, but their relative ordering is preserved. Whereas most existing cell synthesis programs rely on constraints-based symbolic layout compaction systems, in order to generate detailed mask geometries, Picasso introduces an alternative compaction method, for the purpose of layout synthesis. This new compaction approach makes it possible for the system to determine the position as well as the shape of layout objects, during the compaction phase. In a sense, this is equivalent to treating wires as an infinite number of "jogs", all connected in series. Although the compaction method is essentially one-dimensional, the ability to determine the shape of objects at compaction time more than compensates for this apparent limitation.

3.3. DATA REPRESENTATION

Designing an effective data representation scheme for layout synthesis is a real feat. The initial circuit consists of a net list of transistors which contains an explicit description of the connectivity. However, the net list contains very little physical information about the nature of these connections, or the physical attributes of the devices, other than those implied by the definition of a MOS transistor. The final description, on the other hand, contains mainly physical information from which the connectivity can be implied. Each synthesis step applies transformations which introduce new physical constraints based on connectivity information, hence generating an array of intermediate situations.

3.3.1. Symbolic Representation of Layout Objects

The data representation scheme used in Picasso, is an object-based symbolic representation, designed to accommodate the dynamic needs of the complete synthesis process. Symbolic objects are divided into two categories. The first category includes those objects which may be manipulated by the program in symbolic form and hence participate in the topological ordering. Such objects are termed "visible" since each one is associated with a unique symbol used for illustrative purposes throughout this document. The second category of objects is referred to as "invisible" objects, since they are introduced in the database only after the cell topology has been determined. Invisible objects are not manipulated symbolically and only participate in the synthesis of detailed layout geometries. A summary of visible objects, for a single metal process, can be found in Figure 3.3. They include transistors, contact windows and polysilicon wires and ports. Transistors are divided into two groups according to their aspect ratio. Wide transistors are devices for which $W > L$, the most common case. Long transistors ($W < L$) are used as weak devices for special applications. Diffusion

and polysilicon contact objects serve to establish connectivity between the metal layer and underlying chains of transistors. A substrate contact indicates a connection between a power rail and the silicon substrate or a well. Finally, polysilicon wires and ports are used to represent extensions of polysilicon gates for I/O purposes. Note that although most polysilicon wires are invisible objects, wires that introduce a gap in either line of diffusion must be included in the cell topology.

Invisible objects include metal wires and reference points, polysilicon wires and diffusion bridges. The nature of these objects is described in the next sub-section.

OBJECT	SYMBOL
Transistor
Diffusion Contact
Polysilicon Contact
Substrate Contact
Polysilicon Wire
Polysilicon Port

Figure 3.3. Object-based symbolic layout primitives.

3.3.2. Representation of Connectivity

The Picasso database provides three different ways to represent connectivity to accommodate the changing requirements of the problem as the synthesis process progresses. Connectivity among layout objects is represented explicitly at an early point in the synthesis process, and later be implied in the geometry.

The first and most explicit representation of connectivity is available on either the diffusion or the polysilicon layers. Pointers are used among visible layout objects, in effect creating a linked-list structure, for each layer. Note that such pointers convey connectivity information, as well as some degree of physical constraint by specifying which layer should be used, to implement the connection. For example, long chains of diffusion can be represented by linking diffusion contacts and transistors together with diffusion links, indicating that these objects should be merged on the diffusion layer. Similarly, vertically aligned transistors with common gate signals can be linked together with a polysilicon link to indicate that they should be connected with a polysilicon wire. Figure 3.4 shows an example. The structure shown corresponds to a CMOS inverter. Both diffusion chains are comprised of a substrate contact, a transistor and a diffusion contact. The polysilicon link between the two gates, indicates that they are to be connected together by a polysilicon wire.

Figure 3.4. Representation of connectivity via pointers.

The representation of connectivity on the metal layer is a more complex issue. Whereas the polysilicon and diffusion layers are used to connect layout objects that are in physical proximity of each other, metal can be used to connect objects at arbitrary

positions within the cell. Furthermore, it is necessary to express the fact that two objects should be connected with a metal wire, even before their respective positions have been set. Hence, the Picasso database provides two different ways to represent connectivity on the metal layer, as illustrated in Figure 3.5. The most explicit form is a structure called **net-table** shown in Figure 3.5 (a). It describes individual nets as a list of layout objects to be connected. Graphically, the net-table is a two-dimensional structure comprised of n columns and m rows. Each row represents a net whereas columns correspond to virtual grid positions within the cell.

Whereas the net-table contains information about the horizontal span for each net, another representation scheme is needed to construct a vertical ordering of metal nets with overlapping spans. Such a two dimensional topological ordering is the purpose of the routing grid structure shown in Figure 3.5 (b). Rows of the routing grid, now correspond to a physical routing track, defined by the cell structure. Columns, on the other hand, retain their correspondence to virtual grid positions, within the cell.

3.4. IMPLEMENTATION PARADIGMS

Picasso is implemented in Common Lisp. The selection of Lisp for prototype development is based on a number of classic arguments. First, Lisp provides a natural means for representing objects and relationships between them. Dynamic scoping, among other features, facilitates the construction of open databases, easily accessed from arbitrary functions. In that sense, Lisp imposes no penalty for dividing a program into a large number of small functions and hence promotes modularity. Lisp also allows the modeling of procedural knowledge, a vital attribute for synthesis. Finally, Lisp frees the programmer of the tedious tasks of data typing and memory management, hence contributing to faster code generation cycles. Although the ability to use generic functions and data structures has historically resulted in performance

Figure 3.5. Representation of connectivity on the metal layer.

degradation, Lisp compilers now exist that make it possible for programmers to optimize their programs, once the required functionality has been obtained. Recent experiments even show execution times comparable to C programs, based on benchmark programs executed on popular UNIX platforms, such as Apollo and Sun.

Picasso also makes use of the Flavors system, which allows object-oriented programming within the Lisp framework. In fact, Flavors is the cornerstone of the object-based symbolic layout representation used throughout the synthesis process. Each generic object type, listed in Figure 3.3, is associated with a "flavor", for which "instance variables" are defined, as well as "methods" for data manipulation. Individual objects become "instances" of a given flavor with a unique binding of the list of instance variables for that flavor. For example, the "wide_transistor" flavor contains instance variables which specify such parameters as type and aspect ratio, as well as pointers to its neighboring objects on each layer. The "wide_transistor" flavor is also associated with a collection of methods which encompass all procedural knowledge required to determine the shape and the position of a particular instance, based on its physical environment. The Flavors system also supports a simple yet extremely powerful feature, called property inheritance, which makes it possible to define new object types on demand with minimal impact on existing code. Assume, for example, that types of wide transistors must be defined, which are identical in all matters except that one allows forty-five degree bends in the gate geometry whereas the other doesn't. Clearly, all the methods defined for one type are applicable to the other one as well, with the exception of that method responsible for computing gate geometries. This situation can be handled gracefully with a scenario in which a "wide_transistor" flavor is created which contains the methods and instance variables common to both cases. Inheritance allows the creation of more specialized data types on top of this general "wide_transistor", corresponding to the two possible gate structures. These specialized flavors need only contain methods which are specific to each case and may inherit all the procedural knowledge defined for the "wide_transistor" flavor.

Picasso takes full advantage of the property inheritance mechanism, in order to achieve process independence. During the creation of a symbolic layout representation for a given cell, the selection of specific flavors is guided by the qualitative description

of the target process. For example, if the process description specifies that forty-five degree features are allowed in transistor gates, then all the wide transistors present in the cell become instances of the "wide_transistor_45" flavor, the descendent of the "wide_transistor" flavor for which such geometries are allowed. Hence, the layout geometries generated are necessarily consistent with the process description.

3.5. SUMMARY

Picasso is a general purpose leaf cell synthesis system, which is capable of mapping a logic level description into a detailed physical layout. It uses a programmable cell structure based on the line of diffusion layout style, and is currently geared towards static CMOS circuits. The data representation scheme used by Picasso is designed to accommodate the dynamic needs of the complete synthesis process. The selection of Lisp as an implementation paradigm facilitates the construction of an open database as well as the representation of procedural knowledge. Finally, the object-oriented paradigm promotes an advanced form of process independence by supporting certain kinds of qualitative changes in fabrication technology.

CHAPTER 4

SYNTHESIS OF COMPLEX GATES

This chapter describes a heuristic optimization method, used in the Picasso system, for an important class of logic circuits called complex gates. This new approach introduces a number of key improvements over existing systems. In addition to optimal chaining of transistors, through diffusion connections, Picasso also minimizes the number of metal tracks required for routing of internal nets. The restriction that complementary transistor pairs be vertically aligned is relaxed. Hence, optimal chains of transistors may be generated even in the absence of a dual-Euler path in the graph, by crossing carefully selected gate signals. Another advantage resides in the fact that the potential for chaining different cells through Vdd, Vss and output connections is considered in the design space exploration. Finally, the system attempts to exploit the presence of repeated input signals in the logic description.

Two restrictions remain, however, on the nature of synthesized circuits. First, the number of logic levels considered is limited to three (i.e. OR-AND-NOR or AND-OR-NAND). This restriction allows the system to conduct a more thorough search of the remaining design space, without wasting time on cells that are very unlikely to yield an optimal solution. Furthermore, empirical data collected by logic synthesis programs tend to indicate that the return obtained from considering "deeper" logic structures is minimal [DGRS], [KeKL]. The second restriction is on the nature of the gates included in the cell description. These must be limited to four inputs for the system to be guaranteed to produce a solution. This restriction is deemed reasonable since it is normally imposed in manual designs as well, in order to limit the number of transistors in series.

4.1. DEFINITION OF THE GRAPH MODEL

CMOS complex gates can be represented as an n-side and a p-side graph which are each other's duals [UevC]. The n-side graph represents a series-parallel tree of n-type transistors, and the p-side graph, the corresponding p-transistor structure. In the Picasso framework, one of these graphs is called the series-graph since at the high level it is a series collection of sub-graphs. Similarly, the dual of the series-graph is called the parallel-graph. The correspondence between {p-side, n-side} and {series, parallel} depends on the logic structure under consideration. Figure 4.1 summarizes the mapping between the two models. The use of this series-parallel terminology allows the same graph framework to handle both OR-AND-NOR and AND-OR-NAND gates. A distinction between the two structures only occurs when mapping the graph representation into a symbolic layout.

Logic Structure	[UevC]	Picasso
AND-OR-AND-INV	p-side n-side	parallel-graph series-graph
OR-AND-OR-INV	p-side n-side	series-graph parallel-graph

Figure 4.1. Mapping between graph models used in [UevC] and Picasso.

Because the number of logic levels is limited to three, the series-graph in the worst case is a series collection of parallel collections of series-branches. The converse is true for the parallel graph. Hence, the graph representation of any cell can be defined formally as follows:

- A graph G is comprised of a series graph G_s and parallel graph G_p which are each other's duals.
- Series graph G_s is composed of subgraphs $\{G_{ip}: 1 \leq i \leq 4\}$ connected in series. Similarly G_p is the parallel composition of $\{G_{is}: 1 \leq i \leq 4\}$. By definition G_{ip} and G_{is} are each other's duals for all i 's.
- Each graph G_{ip} is the parallel composition of series-branches $\{G_{ijs}: 1 \leq j \leq 4\}$, where each G_{ijs} is the series collection of 0,1,2 or 3 edges. Similarly graph G_{is} is the series composition of parallel-branches $\{G_{ijp}: 1 \leq j \leq 4\}$, where each G_{ijp} is the parallel collection of 0,1,2 or 3 edges.
- A graph is termed "even" if it has an even number of sub-graphs. It is termed "odd" otherwise. Similarly, an even/odd sub-graph has an even/odd number of branches.

Figure 4.2 illustrates the graph representation. The logic structure of Figure 4.2 (a) can be represented as a pair of graphs $\{G_s, G_p\}$, shown in Figure 4.2 (b). The series-graph G_s is the series composition of parallel sub-graphs G_{1p} and G_{2p} which are each comprised of a number of series branches. A similar discussion applies to the parallel-graph G_p .

(a) Logic Description

(b) Hierarchical Decomposition of Graph Representation

Figure 4.2. Graph model example.

4.2. GENERAL ALGORITHM

4.2.1. Graph Traversal Example

This section illustrates the graph-based synthesis method, through the use of a simple example. Figure 4.3 (a) shows a complex gate of type AND-OR-NAND. In order to facilitate the construction of the graph representation from the logic description, each input of the first level gate is associated with a stage. For the example under consideration, the first level gate is a three input NAND gate which yields three stages, labeled as G1, G2, and G3 respectively, in Figure 4.3 (b). A non-optimized graph is constructed by mapping each stage into a pair of dual sub-graphs, as shown in Figure 4.3 (c). For example, stage G1 maps into the pair {G1p, G1s}. G1p is a parallel sub-graph, whereas G1s is its series counterpart. The series-graph is constructed by combining the G1p's in series, whereas the parallel combination of G1s's yields the parallel-graph.

Following the generation of the dual graph representation, a set of heuristics are applied to effect changes to the ordering of sub-graphs and to determine a traversal for each sub-graph. The objective is to find the overall traversal sequence that will produce the best possible layout, in the line of diffusion layout style. The graph generated by Picasso for the sample complex gate is shown in Figure 4.3 (d). Note that the ordering of sub-graphs is changed to G1-G3-G2 and that the order of the branches in each sub-graph has also been changed. The resulting traversal, corresponding to the ordering of gate signals in the layout, is (A B C G H I J K F D E), which is a dual-Euler path, and hence constitutes an optimal solution. The following sub-sections describe the heuristics used to effect these graph transformations.

(c) Graph representation without optimization

(d) Graph traversal produced by proposed approach

Figure 4.3. Example of general synthesis method for complex gates.

4.2.2. Determination of Sub-graph Types

The aim of Picasso's approach is to carefully analyze the nature of sub-graphs associated with each stage, in order to reduce the size of the search space without compromising on optimality. For that purpose, each stage of the logic description is categorized according to two important attributes. The first attribute is based on the parity of the parallel sub-graph G_{ip} , and is determined by simply counting the number of series-branches that compose G_{ip} . For the example shown in Figure 4.3, sub-graphs G_{1p} and G_{2p} are even, since they are both composed of an even number (2) of branches. G_{3p} , on the other hand is odd, since it is comprised of 3 branches. The second attribute is the number of even parallel-branches that compose the series-sub-graph G_{is} . Consider for example, the parallel-graph of Figure 4.3 (c). G_{1s} is comprised of one even parallel-branch (A B) and one odd parallel branch (C). Given these two attributes, all possible sub-graphs can be uniquely assigned to one of ten categories which are summarized in Figure 4.4. Note that the Odd/4 type never occurs given the current restrictions applied to logic circuits covered by this method.

Parity of G_{ip} \ # even branches in G_{is}	0	1	2	3	4
Even	Even/0	Even/1	Even/2	Even/3	Even/4
Odd	Odd/0	Odd/1	Odd/2	Odd/3	Odd/4

Figure 4.4. Summary of sub-graph types.

4.2.3. Graph Traversal Strategy

The key point of the traversal strategy is that all sub-graphs of a given type yield an identical traversal. Thus, in most cases, the inherently difficult complex gate synthesis problem is reduced to a pair of much simpler problems: (i) determine an ordering for the sub-graphs, and (ii) find an efficient traversal for each of the nine possible sub-graph types.

The relative ordering of the sub-graphs is computed based on type information alone. In general it is desirable to include even sub-graphs at the beginning or the end of the order. Traversal of an even sub-graph will normally begin and end at the same point, hence forming a closed loop. On the other hand, infinitely many odd sub-graphs can be traversed in series, since the traversal of an odd number of branches will inevitably begin and end at opposite ends of a sub-graph. If an ordering cannot be found, that will guarantee an optimal traversal, an attempt is made to combine certain sub-graphs, again based on type information alone. Such combinations are discussed in Section 4.3.

The next step consists of determining a traversal strategy for each sub-graph, such that the end point of the traversal for one sub-graph coincides with the beginning of the next one. Since there are only nine possible sub-graph types, it is feasible to determine the corresponding traversal strategies in advance, and to encode them in simple procedures. Figure 4.5 (a) summarizes these simple procedures. In most cases, a slightly different traversal is called for depending on the relative position of the sub-graph in the overall ordering. Therefore, Figure 4.5 (a) distinguishes between the three possible cases. A detailed legend of the graph symbols used in Figure 4.5 (a) can be found in Figure 4.5 (b).

Sub-Graph Type	First Sub-Graph		Middle Sub-Graph		Last Sub-Graph	
	Gip	Gis	Gip	Gis	Gip	Gis
Even/0
Even/1
Even/2
Even/3
Even/4
Odd/0
Odd/1
Odd/2
Odd/3

Figure 4.5 (a). Graph traversal procedures.

Symbol	Number of Branches	Traversal Begins	Traversal Ends
Even	Bottom	Bottom
Even	Top	Top
Odd	Top	Bottom
Odd	Bottom	Top

Figure 4.5 (b). Legend for graph traversal procedures.

Finally, sub-problem solutions are merged to yield the final result, by combining parallel sub-graphs in series, and series sub-graphs in parallel. The series combination of parallel sub-graph is rather straightforward. Continuous chains of transistors are formed as long as the traversal for one sub-graph ends at the same node as the following sub-graphs begins. Otherwise, a "gap" must be introduced in the diffusion line. On the other hand, the parallel combination of series sub-graphs requires that the traversal shown for Gis in Figure 4.5 be mirrored with respect to the x-axis for alternate sub-graphs. Thus, the first series sub-graph is traversed from top to bottom, the second from bottom to top, etc...

4.3. ADVANCED FEATURES

The general synthesis strategy described above is both efficient and simple. Furthermore, it will always produce a "correct" solution. Optimality of the results, however, is not guaranteed. This section presents a number of refinements, that were added to the basic method, in order to ensure optimal results.

4.3.1. Grouping of Sub-graphs

One of the weaknesses of the general method is that it will only find dual-Euler paths that cover each sub-graph in turn. It is unable to detect an optimal path that would alternate between branches of multiple sub-graphs. Hence, in order to achieve optimal results in all cases, it is necessary to introduce the possibility of combining selected sub-graphs, before deciding on a traversal strategy.

Figure 4.6 illustrates a situation where grouping sub-graphs is beneficial. The graph is composed of two stages of type Even/0, and one of type Odd/1. The solution generated by the general method is: (E F C D B A S R) which introduces one gap in the parallel-graph traversal between signals A and S. This constitutes the best possible solution for the traversal of each sub-graph in sequence. However, the approach taken by Picasso consists of first merging stages G2 and G3 into a single "combined" stage. Partial solutions are then produced for the two sub-graphs. The solution for G1 is simply (S R), whereas the combined sub-graph G2/G3 yields the partial traversal (A B E F C D). Hence, the complete traversal is the optimal solution: (S R A B E F C D). The increase in system complexity required to detect and process combined sub-graph is minimal. All dual-Euler paths not detected by the general method involve the combination of exactly two sub-graphs, at least one of which is of type Even/0. Hence, the only required change to the general method to ensure optimal results, is to look for possible combinations of arbitrary sub-graphs with an Even/0 sub-graph.

(a) Sample graph with three stages

(b) Solution produced by general method

(c) Solution produced by grouping G2 and G3

Figure 4.6. Optimal solution found by grouping sub-graphs

4.3.2. Splitting of Gate Signals

One of the key advantages of Picasso's approach over existing systems is the removal of the restriction that complementary transistor pairs be vertically aligned, hence allowing certain gate signals to be crossed. There are two cases where this yields significant benefits. If both the series-graph and the parallel-graph possess Euler paths but fail to have one in common, then crossing carefully selected gate signals may produce an optimal chaining of transistors, without otherwise increasing the cell area. Furthermore, even if a dual-Euler path exists, some other objective may be met by re-ordering gate signals.

Figure 4.7 illustrates the benefits of crossing gate signals to achieve optimal chaining. A dual-Euler path is made impossible by the presence of three even stages in the series graph. Breaking gate signal D into two segments, however, results in two optimal chains of transistors. Completing net D with a horizontal metal line does not result in increased cell area, since the separation between p-type and n-type devices required by physical design rules is usually sufficient to allow one or two metal tracks at the center of the cell.

Figure 4.8 illustrates the use of gate splitting to achieve optimal chaining with adjacent cells. Figure 4.8 (a) shows a graph traversal using the obvious dual-Euler path solution. The nature of the signals present at both ends of the resulting outline makes it impossible to chain its diffusion lines with adjacent gates. Figure 4.8 (b), on the other hand, shows a graph traversal using gate splitting to yield Vdd and Vss connections at both ends of the cell. Crossing gate signals D and C, and completing net B with a horizontal metal line, again, does not result in increased cell area. On the other hand, Vdd and Vss connections at both ends of the cell increase the potential for chaining with adjacent gates. Again, specific uses of gate splitting are limited to combinations with Even/O sub-graphs.

(a) Gate logic description

(b) Graph traversal

(c) Outline of the layout

Figure 4.7. Use of gate-splitting to achieve optimal chaining.

(a) Graph traversal produced by following dual-Euler path

(b) Graph traversal produced using gate splitting

in order to obtain V_{dd} and V_{ss} connections at both ends of the cell.

Figure 4.8. Use of gate-splitting for chaining with adjacent cells.

4.3.3. Analysis of Input Signals

In addition to splitting of gate signals, appropriate handling of repeated input signals may produce an optimal solution for some graphs. Such is the case for the graph of Figure 4.9, which would appear not to possess a dual-Euler path, again due to the presence of three even stages in the series graph. However, detecting the fact that input signal D appears in both G2 and G3 allows the combination of both G2 and G3 into a single sub-graph, which can be traversed in the sequence (C D E D), and hence, yield the optimal solution (B A C D E D) for the overall graph.

In practical terms, observing repeated input signals, can be applied to a limited number of cases. Specifically, repeated input signals can serve as a reason to combine a number of Even/0 sub-graphs or can help find optimal solutions for individual sub-graphs of type Even/2, Odd/2, Even/3, Odd/3 and Even/4.

Figure 4.9. Use of repeated input signals in graph traversal.

4.3.4. Interconnect Considerations

One of the side effects of grouping sub-graphs is a potential increase in the number of metal tracks required in order to complete internal routing. For that reason, Picasso makes it possible for the user to specify an upper bound on the number of metal tracks for each region of the cell. Imposing such a restriction results in some rules being excluded during the initial search for sub-graph combinations.

In addition, care must be taken when assigning diffusion signals to graph nodes, to select a mapping which generates the smallest possible number of metal tracks for routing of internal nets. Indeed, following the determination of a graph traversal order, the top and bottom nodes of each graph correspond to a power rail signal and the output signal. Which graph node gets mapped onto which diffusion signal does not affect the logic behavior of the circuit, but can have a dramatic impact on interconnect.

Figure 4.10 shows an example. The circuit shown in Figure 4.10 (a) was used as an example in [MaHa]. Whereas Maziasz and Hayes claim optimality in terms of diffusion chaining, a clear improvement over earlier work, Picasso produces a solution which is also optimal in the number of metal tracks required for routing of internal metal nets. The dual-Euler path selected yields the gate ordering (A B C D G H F E), shown in Figure 4.10 (b). Nodes 1, 2, 3, 4, 5, 6 and 7 have to be completed with metal connections that should be internal to the cell. Part (c) of the Figure illustrates the span of each net in order to help determine which nets can share a single routing track inside each region. It is clear that net 1 can share a routing track with either net 3 or net 4, and that no other combinations are possible. Hence, if net 4 is assigned to be the Vdd rail, then nets 1 and 3 can share a track, as shown in Figure 4.10 (d). On the other hand, if net 1 is assigned to be the Vdd rail, then no combinations are possible and an additional metal track is required to complete internal routing. Figure 4.10 (e) shows the results produced by [MaHa] for the same example.

(a) Gate logic description

(b) Graph traversal

(c) Net span analysis

(d) Outline of the layout

(e) Layout produced by [MaHa]

Figure 4.10. Assigning graph nodes to diffusion signals.

4.4. SUMMARY

Picasso uses a heuristic approach for the synthesis of complex gates, which is based on a graph-theoretical circuit representation. The basic method consists of mapping each "stage" of the logic circuit into a dual sub-graph, associated with a unique type identifier. The traversal of each sub-graph as well as the ordering of sub-graphs are determined based on type information alone. In order to ensure optimal results within the specified restrictions, a number of advanced features were added to the general synthesis method. These include the ability to combine sub-graphs to achieve optimal chaining, crossing carefully selected gate signals and exploiting repeated inputs. Finally, the system also minimizes the amount of interconnect required to complete the implementation of internal nets.

CHAPTER 5

TOPOLOGICAL CELL EXPANSION

This chapter describes the necessary operations to transform a transistor level net list into a complete symbolic layout representation, in preparation for layout synthesis. Recall that the result of transistor level design is a transistor structure with an associated internal ordering for each cluster. Therefore, transistor placement can be completed by defining a relative ordering for the clusters, by permutation and mirroring alone, without changing the transistor structures associated with each cluster. The routing problem which arises at this point differs significantly from the channel routing problem. The points to be connected may appear anywhere within the routing region, and are not restricted to the edge of a routing "channel". The effect of the routing algorithm on the database is two-fold. First, it generates a loose wiring for the cell, based on a two-dimensional virtual grid. But, in order for metal wires to be properly aligned with underlying objects, the routing algorithm must also impose constraints on the vertical position of objects in the symbolic description. To ensure that such constraints are feasible, the nature and physical attributes of these objects are considered during the routing phase.

5.1. CLUSTER PLACEMENT

The cluster placement problem consists of defining a one-dimensional topological ordering as well as an orientation for the clusters. Given a cell with n clusters, a single placement must be selected out of the $n!2^n$ possible combinations. Because Picasso generates fixed-pitch cells, the objective is to minimize the length of the cell while insuring that all the internal connections can be completed, in the vertical space available.

Although common diffusion connections have a general tendency to reduce the amount of metal interconnect inside a cell, optimal chaining alone is not a sufficient metric to evaluate possible placements. On the other hand, generating a placement based solely on interconnect, while ignoring the potential for chaining through common diffusions is also unacceptable. The cluster placement technique used by Picasso takes advantage of both metrics by defining an "affinity" function, which combines the two criteria. The affinity function reflects the nature of the electrical connections between the clusters as well as their potential for chaining through common diffusions, based on morphology. The placement method also allows the weight attributed to each metric to be fine tuned according to the cell structure being used.

5.1.1. Representation of Cluster Morphology

Cluster morphology is represented internally using a tile-based method illustrated in Figure 5.1. Each cluster is associated with one tile, which contains the names of the signals present on the diffusion layer at the extremities of the cluster. The four corner sections of each tile are associated with diffusion signals in their respective positions. For example the symbol in the upper left hand corner is the signal carried by the diffusion contact in the p-region, on the left edge of the cell. The tiles also include lists of input and output signals useful in evaluating interconnect.

Figure 5.1. Tile representation for clusters.

5.1.2. The Connectivity Graph

A data structure called the "connectivity graph" is used to represent connectivity between clusters. A cluster x is mapped onto a vertex V_x in the graph, and the weights assigned to non-directed graph edges reflect the degree of connectivity between clusters. For the purpose of computing weights, connections between clusters are first assigned a unique identifier based on the following definitions:

- **Internal Connections** - An internal connection corresponds to a signal that appears in the list of inputs for one cluster and the list of outputs for another. Alternatively, an internal signal may appear in the output list of two different clusters.
- **Type-1 Connections** - A type-1 connection is an internal connection associated with a signal that is not a cell-level output.
- **Type-2 Connections** - A type-2 connection is an internal connection associated with a signal that appears in the list of cell outputs.
- **Type-3 Connections** - A type-3 connection represents cell-level inputs connected to more than one cluster.

The weight of an edge between two clusters x and y is defined as:

$$E(x,y) = (n1*w1) + (n2*w2) + (n3*w3), \quad (5.1.1)$$

where $n1$, $n2$ and $n3$ are the number of type-1, type-2 and type-3 connections respectively, and $w1$, $w2$, and $w3$ are the weights assigned to each type.

Type-1 connections are normally assigned a relatively heavy weight, since they are likely to connect all three cell regions. A lighter weight is assigned to type-2 connections due to their amenability to an external implementation. Indeed, since cell outputs are very likely to occupy a track in a routing region adjacent to the cell, implementing a portion of such connections outside the cell may represent a minimal area overhead. Finally, type-3 connections are assigned the lightest weight for two reasons. The first is that common input connections are less likely to occupy the center of the

cell and hence, may cover long distances in either the p-region or the n-region. The other reason is that should the metal connections not all be feasible simultaneously, connections between repeated inputs are the first to be considered for an out-of-cell implementation.

5.1.3. Affinity Between Clusters

The affinity $A(x,y)$ between two clusters x and y , reflects the likelihood of the two clusters to form common diffusion connections, as well as the degree of connectivity between them. It is defined as

$$A(x,y) = A_m(x,y) * W_m + A_c(x,y) * W_c, \quad (5.1.2)$$

where $A_m(x,y)$ is the affinity function based on morphology, and $A_c(x,y)$ is the affinity function based on connectivity. The parameters W_m and W_c are user-defined weighting factors.

The morphology-based component of the affinity function, $A_m(x,y)$, is defined as the likelihood that two clusters will abut naturally through common diffusion connections, should they be placed side-by-side in the layout, and given a random orientation. Since we are considering clusters two at a time and there are two possible orientations for each cluster, a total of four possibilities must be investigated. Furthermore, diffusion connections are possible in both the p-region and the n-region, resulting in a total of eight potential connections. For example, for the two clusters shown in Figure 5.2 (a), the eight comparisons listed in Figure 5.2 (b) are carried out. The value of $A_m(x,y)$ is increased by one whenever the signals being compared are identical. Hence, $A_m(x,y)$ is an integer between zero and eight.

The connectivity-based component of the affinity function, $A_c(x,y)$, is defined as:

SIG-A	x	SIG-C	SIG-E	y	SIG-G
	(input signals)			(input signals)	
SIG-B	(output signals)	SIG-D	SIG-F	(output signals)	SIG-H

(a) Example clusters

P_REGION	SIG-A with SIG-E
	SIG-A with SIG-G
	SIG-C with SIG-E
	SIG-C with SIG-G
N_REGION	SIG-B with SIG-F
	SIG-B with SIG-H
	SIG-D with SIG-F
	SIG-D with SIG-H

(b) List of comparisons

Figure 5.2. Computation of morphology-based affinity.

$$Ac(x,y) = E(x,y) + a\{E(x,u)+E(u,y)\}, \quad (5.1.3)$$

where $E(x,y)$ is the weight of the edge from V_x to V_y in the connection graph. The second member in the equation is a second order connection component that represents connections between x and y through a third cluster u . Experiment has shown that a third order component does not yield any improvement in the resulting placement for problems comprised of less than 10 clusters and unduly increases the complexity of the algorithm. Empirical results show that the value of the parameter "a" should be less than 0.3, for first order connections to prevail in the calculation of $Ac(x,y)$.

Finally, the parameters W_m and W_c are weighting factors used to fine tune the performance of the system to specific requirements. In particular, the target cell structure should have a significant impact on the selection of these parameters. For fixed pitch cells, with limited area for interconnect, the weight attributed to each criterion should be such as to put the emphasis on interconnect minimization rather than the optimization of chaining through common diffusion connections. The function $A_m(x,y)$ should be used to resolve ties resulting from the applications of the other two functions, and to direct the selection of cluster orientation. On the other hand, if unlimited resources exist for interconnect, a slight bias in favor of morphology may produce better results. However, attributing a large relative weighting to morphology yields a greedy and very near-sighted placement metric.

5.1.4. Placement Algorithm

Following the above definition of cluster affinity based on morphology and connectivity, two n by n tables can be constructed representing the affinity of each pair of clusters based on either criteria. A total affinity figure is then computed for each cluster, which is the sum of all the entries in both tables for that cluster. The cluster with the highest total affinity figure is placed first, and becomes the only element of the "set of placed clusters." In essence, this first cluster can be viewed as a seed, around which all the other clusters will be placed.

The next cluster to be placed is selected by computing the affinity of each remaining cluster to the "set of placed clusters." The cluster with the highest total affinity to placed clusters then becomes the second placed cluster. That process is repeated until no unplaced clusters remain.

The selection of a placement for a new cluster, relative to the set of placed clusters, is performed by selecting one of eight possible cases. The first four alternatives

are self evident. The new cluster can be placed at either end of the chain and with either one of two possible orientations. The other four alternatives stem from the fact that Picasso also allows the placed cluster at either end of the chain, to be mirrored with respect to the y-axis, if such a move produces a better placement solution. Figure 5.3 summarizes the eight solutions considered.

The merit of each possible placement is evaluated, again by considering the two placement metrics of morphology and connectivity. However, the affinity based on morphology can now be evaluated exactly. This parameter contributes a positive value of either 0, 1 or 2. Connectivity information, on the other hand, is used to compute the detrimental effect of the candidate placements on internal interconnect. For each element of the set of placed clusters, the weight of the edge $E(x,y)$ in the connectivity graph, between that cluster and the new cluster under investigation, multiplied by the distance between the two clusters in the candidate placement, contributes a negative value to the merit function.

(a) Current set of placed clusters

(b) Eight possible placements for new cluster

Figure 5.3. Placement selection for new cluster.

5.2. ROUTING OF INTERNAL NETS

Following cluster placement, each signal net which has not been completely implemented via common diffusion connections or a vertical polysilicon wire, should be extracted from the symbolic layout and included in a net-table, in preparation for the routing phase. Both the symbolic representation of the diffusion chains and the net-table are supplied as inputs to the routing algorithm.

5.2.1. Routing Regions

Recall that the layout style used by Picasso allows an almost complete overlap of the routing and active areas. For that reason, the routing area is divided into three distinct regions, corresponding to the three regions defined by the cell structure: p-region, n-region and center-region. The number of metal tracks allowed in either region is a function of the cell structure parameters, defined by the user. However, a minimum of two metal tracks, in each region, is required by the routing algorithm, a restriction often imposed in manual layouts as well.

5.2.2. Net Types

The routing algorithm distinguishes between nets that can be completely implemented in a single routing region from those that need to cross between regions. Hence, four different net types are defined: (i) p-type, (ii) n-type, and (iii) center-type, for nets restricted to a single routing region, and finally (iv) cross-type, for nets that cover different regions. The reason for this differentiation by net types is simple. Nets which are restricted to a single region can normally be implemented with a single horizontal wire. These nets very seldom interfere with one another. Cross-type nets, on the other hand, will inevitably require both vertical and horizontal segments, hence causing blockages for other horizontal nets.

5.2.3. Assignment of Nets to Regions

The routing algorithm attempts to reduce the complexity of the task at hand, by first decomposing it into three loosely coupled sub-problems. Each sub-problem deals with a single routing region, involves fewer nets, and presents fewer choices. This problem partitioning is achieved by assigning complete nets, or alternatively, segments of nets, to individual routing regions. Data representation is achieved through the use of three net tables, each associated with a unique region. Region assignment is carried out by transferring information from the main net-table to region net-tables. The assignment of n-type and p-type nets to routing regions is self evident. These are handled first, and simply assigned to their respective region. The treatment of cross-type and center-type nets, however, is more subtle.

The philosophy for cross-type nets consists of first determining the vertical position at which a net should traverse horizontal region boundaries. Cross-type nets are thus reduced to a number of smaller net segments of one of the other three types. This operation is performed by considering the nature and the range of each cross-type net. Priority is given to the net for which the least freedom exists. The crossing position must be determined in such a way, that sufficient room exists in each region to accommodate the new net over the specified range. The center region is given preferential treatment, since the success or failure of the routing algorithm hinges upon a judicious exploitation of that most precious resource. Thus, the primary concern is to reduce the range covered by each net, in that region, to a strict minimum.

Center-type nets are handled last. Given that such nets are most often connections between two polysilicon wires, it may be possible for some of them to be partially assigned to either the p-region or the n-region in order to avoid blockages caused by vertical segments of cross-type nets. Not only are these polysilicon to polysilicon connections of the least critical kind, but there may not be any obligation to keep them in

the center if there exists sufficient room in the other cell regions to implement contact windows to polysilicon wires. In most cases, if the cell structure is consistent with the internal routing requirements, it should be possible to bind all the net segments to a routing region. However, some nets may remain unassigned until a later step discussed in Section 5.2.5.

5.2.4. Ordering Nets within Regions

Once all the nets have been broken down into segments, each associated with a distinct routing region, a topological ordering of nets within each region is determined. Vertical constraints are generated and are used to resolve conflicts. For example, an attempt is made to assign the segments of cross-type nets that run in either the p-region or the n-region as close to the center as possible. On the other hand, p-type and n-type nets are typically positioned as far from the center as possible.

5.2.5. Topological Compaction

All the steps of the routing algorithm to this point, have attempted to restrict the use of the center region, as much as possible. Whereas the aim of this strategy is to ensure the feasibility of the maximum number of nets, given that the center of the cell constitutes the main bottleneck, such a conservative approach may introduce blockages in either the p-region or the n-region, which are detrimental to the overall objective. This situation can be remedied by a simple post-processing step, which consists of forcing all the nets to gravitate towards the center of the cell. This operation, called topological compaction, allows some nets running in the p-region or the n-region to be partially transferred to the center-region. The routing space made available in this fashion, may be used to implement unassigned nets. Failure to assign all the nets to specific routing tracks, at this point, results in the failure of the overall routing process.

5.2.6. Creation of Routing Grid and Back Annotation of Symbolic Layout

Finally, the resulting two-dimensional ordering for metal wires is represented using a single routing grid, which replaces all the routing tables in the database. Connectivity on the metal layer, which was represented explicitly up to this point, is now completely embedded in the geometry.

In addition to its primary purpose of determining a two-dimensional ordering of metal connections within the cell, the routing algorithm determines the horizontal positions of the source and drain connections for transistors. Polysilicon contacts must also be added to reflect the positions of metal wires within the cell.

5.2.7. Routing Example

A simple example, shown in Figure 5.4, will serve to illustrate the various steps of the routing algorithm. Figure 5.4 (a) contains the initial symbolic layout and net-table. Squares and circles represent required connections to the diffusion and polysilicon layers, respectively. The position of symbols with respect to horizontal lines indicate the cell region where the connection should be made. For this example, the net-table contains three "cross-type" nets, labeled A, B and D. The crossing position of each net is determined, starting with net B, since it is the net for which the least freedom exists. The crossing positions are 1, 5 and 10 for nets A, B and D, respectively. Figure 5.4 (b) shows the result of assigning nets or net segments to individual routing regions. This assignment step would normally be followed by a topological ordering of net segments within each region. For such a simple example, however, this constitutes a trivial operation. The mapping of net segments on a two-dimensional routing grid is shown in Figure 5.4 (c), whereas Figure 5.4 (d) contains the back annotated version of the symbolic layout representation.

(a) Symbolic layout with net table

(b) Assignment of net segments to routing regions

Figure 5.4 Simple Routing Example

(c) Two dimensional routing grid

(d) Back annotated symbolic representation

Figure 5.4 Simple Routing Example (continued)

5.3. COUPLING BETWEEN PLACEMENT AND ROUTING

One of the key requirements of cell compilation systems is to always produce a solution, regardless of the design constraints. With that objective in mind, each segment of each net present in the cell description is assigned a priority figure, before the cluster placement phase begins. In the event where it is impossible to route all the metal nets inside the cell area, the cluster placement process is repeated after the lowest priority net segment has been removed from the routing table and assigned as external to the cell. Hence, the overall topological expansion mechanism may be summarized as follows:

- Decomposition of nets in segments and assignment of priority figures.
- Cluster placement:
 1. Build connectivity graph and compute affinity tables.
 2. Select first cluster based on highest affinity to other clusters.
 3. While clusters remain; select cluster based on affinity to placed clusters, and choose placement with highest merit, for that cluster.
- Routing of internal nets:
 1. Determine the type of each net segment.
 2. Assign net segments to routing regions. Attempt to minimize the use of the center-region.
 3. Determine ordering for nets within regions.
 4. Move nets towards the center of the cell. Attempt to free up some space in the p-region and n-region, to assign remaining nets, if any. If some nets remain unassigned, the routing phase fails.
 5. Otherwise, create routing grid, in preparation for detailed layout synthesis.
- If routing failed, remove one net segment from the net list, with the lowest priority figure, and repeat the placement and routing operations. Otherwise, topological expansion is complete.

5.4. SYMBOLIC LAYOUT REFINEMENT

Following the device level placement and routing, a number of simple operations must be performed on the symbolic layout representation. In particular, I/O port positions must be established for each external signal. Because the compaction method used by Picasso is one-dimensional, the horizontal position of transistors as well as the orientation for substrate contacts must also be pre-determined.

5.4.1. Implementation of I/O Ports

Since the current implementation of Picasso only supports a single metal process, I/O ports are implemented on the polysilicon layer, by extending gate wires to the top and bottom edges of the cell. The default assignment of port positions is based on the first occurrence of a signal, from left to right, in either cell region. However, Picasso makes it possible for the user to specify a number of two-variable relations of the form "A before B", which specify the preferred relative positions of two I/O signals. If such a constraint is applied and a choice exists for either signal, the system attempts to find an assignment that satisfies the constraint without otherwise modifying the topology of the circuit. Thus, if both signals appear at only one position within the cell, or if a constraint is not feasible for any other reason, the situation is reported to the user, and the constraint is ignored.

The inability of the system to ensure that preferred I/O positions are always obeyed could become a limitation for certain applications. Since the potential solutions to this problem have a direct impact on a number of different modules, their treatment is postponed until Chapter 9.

5.4.2. Horizontal Position of Transistors

The horizontal position of transistors is determined by applying simple heuristics. The size of the transistor is considered as well as its position within a chain of devices, and the horizontal position of diffusion contacts to its source or drain region. Furthermore, provisions may have to be made for polysilicon contacts to the gate wire within the active region.

5.4.3. Orientation for Substrate Contacts

As mentioned earlier, all diffusion connections to a power rail are implemented as "split contacts", in order to ensure the presence of a substrate contact for each V_{dd} and V_{ss} connection. These substrate contacts can have one of three possible orientations, as will be discussed in Chapter 7. The orientation for individual contacts is again determined heuristically, based on the nature and the size of the neighboring devices.

5.5. SUMMARY

The topological expansion process consists of cluster placement and routing of internal nets. Placement is based on a composite metric which reflects connectivity among the clusters as well as their amenability to common diffusion connections. The routing algorithm first assigns net segments to routing regions, followed by a topological ordering of segments within each region. The system may limit the height of the cells by allocating net segments as external to the cell.

CHAPTER 6

FLEXIBLE BOUNDARIES FOR LAYOUT SYNTHESIS

Layout synthesis, or the generation of final mask geometries associated with each cell, is by far the most complex task of cell synthesis. The human experience will attest to that. Indeed, even simple designs go through numerous iterations and often get passed from one designer to another for verification before being released. Whereas two-dimensional compaction of symbolic layouts yields significant speed-ups for custom cell design, a penalty in terms of compactness seems inevitable. Devices and wires are "fleshed" to their final shapes and sizes before compaction begins, except for possible insertion of "jogs" in selected wires. No effort is made to adapt the shape of layout objects to that of their environment. Picasso uses a new symbolic layout compaction method, in which the very shape of layout objects added to the circuit is determined by the nature of those objects already present in the circuit. Its operation is based on the definition of forbidden zones, associated with each layer, and represented as flexible boundaries. Flexible boundaries bear a strong similarity to "contours" used in the context of river routing [Ullm] [Hsu], and compacted channel routing [XiKu] [Deut] [RPVN], used to determine the minimum spacing between wires on a single layer. In this thesis, however, the concept is extended to support minimum dimension and overlap rules, in addition to minimum spacing. Picasso's algorithm also considers the strong interaction between the different layers to compute the shape of objects.

6.1. THE CONCEPT OF FLEXIBLE BOUNDARIES

A flexible boundary is a data structure used to represent a "forbidden zone" created by layout objects on a particular layer. A single object may be associated with a number of boundaries each serving a different purpose. For example, a polysilicon wire can set boundaries on the polysilicon layer which will constrain other polysilicon objects while at the same time setting a second boundary for layout objects on the diffusion layer.

The principal advantage of flexible boundaries over constraint-graphs resides in the fact that they can be used to determine the shape of layout objects directly, without requiring the explicit insertion of "jogs". In a sense, this new compaction method treats wires as an infinite number of "jogs", all connected in series, which makes it possible for layout objects to adopt the exact shape of their constraining elements. Earlier concepts used in river routing [Ullm] and compacted channel routing [Deut] are extended to address the complete CMOS layout problem. Figure 6.1 illustrates that concept by showing a simplified version of the boundaries created by a polysilicon wire, labeled as W-1. Boundary B-1 is exactly one poly-poly spacing away from W-1 and represents the extent of the forbidden zone for other polysilicon objects. Boundary B-2, on the other hand, represents the forbidden zone for layout objects on the diffusion layer, and its spacing from the edge of W-1 corresponds to the poly-diffusion spacing in the design rules. These flexible boundaries can be used in a number of ways. Figure 6.1 (b) shows how boundary B-2 can serve to determine the position of a new layout object on the diffusion layer, whose shape and vertical position have been pre-determined. On the other hand, in Figure 6.1 (c), boundary B-1 is used as a guide to determine the actual shape of polysilicon wire W-2.

(a) Layout Object and Associated Boundaries

(b) Determining the Position of Objects

(c) Wire Generation

LEGEND

Polysilicon

Diffusion

Polysilicon Boundary

Diffusion Boundary

Figure 6.1. Flexible boundary concept

6.2. IMPLEMENTATION OF FLEXIBLE BOUNDARIES

In order to use flexible boundaries for layout synthesis, a number of basic operations must be defined. In particular, it is necessary to determine how new segments can be added to a boundary definition when objects are introduced in the layout, and what the nature of these new segments should be, in order to effect the appropriate shift in boundary position. This section first addresses issues of data representation and then describes some simple operations for boundary manipulation.

Although flexible boundaries constitute a general tool, and can be applied to various technologies, this thesis illustrates the use of flexible boundaries in the context of the Picasso system, which builds cell geometries by first performing a two-dimensional topological ordering of layout objects followed by a one-dimensional layout compaction along the horizontal direction. Hence, boundaries traverse the cell area from bottom to top, dividing the cell into two regions: a "forbidden" region, to the left of the boundary, and an "allowed" region, to the right.

6.2.1. Data Representation

A flexible boundary, in the Picasso framework, consists of a set of segments forming a continuous line. It is represented as a list of vertices, joining these segments, in order of ascending y-coordinates. Figure 6.2 illustrates the representation scheme. Figure 6.2 (a) shows the graphical representation, whereas Figure 6.2 (b) shows the internal machine representation as a list of vertices. Note that, the only line segments allowed in a boundary definition are vertical, horizontal or at a 45 degree angle with the horizontal. Furthermore, y-coordinates of vertices must be increasing monotonically.

(a) Graphical Boundary Representation

(5 0) (5 3) (0 3) (0 7) (2 9) (6 9) (6 13) (4 13) (4 17)

(b) Internal Boundary Representation

Figure 6.2. Flexible boundary representation

6.2.2. Updating Boundaries

Adding objects to the layout causes the forbidden zones for certain layers to expand. This must be reflected in a shift of the associated boundary towards the right, which can be effected by adding segments to the appropriate boundary definition. Figure 6.3 illustrates the process of adding a segment to a boundary definition. Only that part of the new segment which lies to the right of the existing boundary causes a shift of that boundary, hence expanding the forbidden zone for the associated layer. The steps involved in this operation can be summarized as follows:

- Find the relevant sub-boundary - Find a sub-boundary comprised of only those segments whose vertical range intersect the vertical range of the segment to be added. For the example of Figure 6.3, the relevant sub-boundary for segment (S T) is the sub-boundary (A B C D).
- Decompose sub-boundary into pairs of points - Sub-boundary (A B C D) is decomposed into the list of pairs ((A B) (B C) (C D)).
- For each pair do:
 - Find relevant sub-segment - A relevant sub-segment is that part of the segment to be added which intersects the vertical range of a given pair. For the pair (A B), the relevant sub-segment is the sub-segment (S U). Figure 6.3 (b) summarizes the results obtained for each pair.
 - Compute new sub-boundary piece - A new sub-boundary piece is computed by augmenting each pair with the part of the corresponding relevant sub-segment which lies to its right. The pair (A B) yields the new sub-boundary piece (A P S U).
- Merge boundary pieces - Compute the new sub-boundary by merging the results obtained for each pair.

(a) Graphical Representation

PAIR	RELEVANT SUB-SEGMENT	SUPERSEEDING PART OF SEGMENT	SUB-BOUNDARY PIECE
(A B)	(S U)	(S U)	(A P S U)
(B C)	(U U)	(U U)	0
(C D)	(U T)	(V T)	(C V T Q D)

(b) Summary of Operations

Figure 6.3. Adding segments to a boundary definition.

The nature of the segments to be added to a boundary definition following the addition of a new layout object, depends on the shape of that object. Two distinct cases must be considered, as shown in Figure 6.4. Three segments are required in the case of a vertical edge. On the other hand, Figure 6.4 (b) shows the single segment required by the inclusion in the layout of an object with an oblique edge.

(a) Adjust Boundary from Vertical Edge

(b) Adjust Boundary from Oblique Edge

Figure 6.4. Determination of the nature of segments to be added.

6.3. DETERMINING THE POSITION OF OBJECTS

The first use of flexible boundaries in layout synthesis consists of finding the left-most position of an object whose shape and vertical position have been pre-determined. Figure 6.5 illustrates the solution to that particular problem. For a layout object which is implemented on a single layer, the left-most position is computed by finding the maximum x-value of the sub-boundary corresponding to the vertical range of the object. The steps involved in this operation can be summarized as follows:

- Find the relevant sub-boundary - This process is analogous to that described above for adding segments and yields the sub-boundary (A B C), for the example of Figure 6.5
- Decompose sub-boundary into pairs - The decomposition of the relevant sub-boundary into pairs yields the list of pairs ((A B) (B C)).
- For each pair do:
 - Find relevant sub-pair - Figure 6.5 summarizes the results obtained for each pair.
 - Compute maximum x-value of relevant sub-pair - The maximum x-value of a sub-segment is easily found by determining the maximum value of the x-coordinates of its vertices. Again, Figure 6.5 summarizes the results obtained for each pair.
- Compute maximum x-value of relevant sub-boundary - The maximum of boundary (P B Q) is at point Q(2,6). Hence, the left-most position of segment (R S) is 2.

Complications may occur, however, in the case of objects, for which layout geometries are defined on multiple layers. In such cases, boundaries on individual layers must be considered in turn, as well as the interaction between the different layers.

(a) Graphical Representation

PAIR	RELEVANT SUB-PAIR	MAXIMUM X-VALUE
(A B)	(P B)	0
(B C)	(B Q)	2

(b) Summary of Operations

Figure 6.5. Determining the position of an object of fixed shape.

2 of/de 2

MicroD

6.4. DETERMINING THE SHAPE OF MOS TRANSISTORS

Determining the shape of layout objects based on boundary definitions is a more complex process, which may involve the use of a number of different boundaries as well as the analysis of additional constraints specific to individual objects.

In determining the shape and position of an MOS transistor, boundaries affecting both the polysilicon layer and the diffusion layer have to be considered, while insuring that design rules regarding spacings and overlaps are also satisfied. Consider, for example, the merger of two transistors which are aligned horizontally, as depicted in Figure 6.6 (a). In such a case, the boundary definitions associated with the diffusion layer can be ignored. The gate boundary alone can be used to determine the shape of the next device.

On the other hand, physical constraints may prevent the alignment of transistors to be merged. As shown in Figure 6.6 (b), a number of different boundaries must now be considered. In particular, the final position of the diffusion region for the new transistor is obtained from a boundary on the diffusion layer that was set to ensure proper spacing from the existing transistor gate. Also, the shape of the new transistor gate is now determined by a boundary on the polysilicon layer to ensure proper spacing from the existing transistor's diffusion region.

(a) Merging transistors with coinciding active ranges

(b) Merging transistors with offset active ranges

LEGEND

Figure 6.6. Determining the shape of MOS transistors

6.5. DETERMINING THE SHAPE OF WIRES

6.5.1. General Wire Generation Algorithm

Using boundaries directly to determine the shape of layout objects can lead to rather bizarre situations. In particular, it can cause useless dents in wires, which can ultimately introduce design rule violations. In order to prevent this phenomenon, filter functions must be defined to eliminate undesirable bends in a boundary definition before it is used to generate geometries. Figure 6.7 illustrates the purpose of a filter function used for wire generation. The filter simply adds a few segments to the existing boundary definition, before it is used as a guide to generate a wire.

Figure 6.7. Sample filter function for the generation of wires.

A number of filtering strategies are possible to accommodate the specific needs of different applications. The basic algorithm consists of recognizing a number of specific patterns in the boundary definition, and determining the nature of the segments to be added based on the patterns found. Figure 6.8 illustrates the set of patterns used.

Solid lines in the figure represent these patterns whereas the dashed lines represent corrective segments. For example a filter function aimed at the removal of forty-five degree bends, could make use of patterns A to F. Furthermore, a maximum reach parameter can be specified for each pattern so that large jogs which can be useful for compaction may remain and other ones removed. Finally, a filter function to smoothen out boundaries, and replace ninety degree bends by one-hundred and thirty five degree bends, could include patterns H and I. Such a filter is particularly useful for transistor gates, in order to facilitate the extraction of electrical parameters from the layout.

Following the boundary filtering process, the wire generation algorithm appears rather trivial in comparison. It consists of a one to one mapping of the remaining boundary segments by corresponding "wire segments" shown in Figure 6.9.

6.5.2. Generation of wires constrained on both sides

The wire generation method described in the previous section is very efficient in terms of area utilization, but can sometimes produce erroneous results due to excessive greed. Consider, for example, the simple case shown in Figure 6.10 (a). A contact window is shown with its associated polysilicon cover, as well a simple boundary to be used as a guide for the generation of a polysilicon wire originating from that contact window. A wire generated with the boundary as specified causes a polysilicon spacing rule violation, which is very hard to predict a priori.

Figure 6.10 (b) illustrates the method used by Picasso, in order to alleviate such trivial, yet catastrophic situations. The initial boundary definition is first shifted to the right, a distance corresponding to the width of the polysilicon wire projected. The "shifted boundary" is then filtered to remove geometries that could result in a design rule violation on the right edge of the wire. The resulting "filtered shifted boundary" is then shifted back by the same amount and used directly to generate the required wire.

Figure 6.8 Patterns used for boundary filtering

Figure 6.9 Wire segments corresponding to boundary segments

(a) Potential Problem with Flexible Boundaries

(b) Shifted Boundary to Avoid Violation

LEGEND

Figure 6.10 Generation of wires constrained on both sides

6.6. SUMMARY

A flexible boundary is a data structure used to represent physical constraints among layout objects. Flexible boundaries can be used to determine the position as well as the shape of layout objects. In the Picasso framework, boundaries traverse the cell area from bottom to top, dividing the cell into "allowed" and "forbidden" regions. Efficient algorithms have been developed for various boundary manipulation operations. These include a number of filter functions to eliminate redundant jogs from the final layout geometries.

CHAPTER 7

CMOS LAYOUT SYNTHESIS USING FLEXIBLE BOUNDARIES

This chapter describes the symbolic layout compaction mechanism which is at the heart of Picasso's layout synthesis module. The first section outlines the nature of the flexible boundaries required to generate layout geometries for CMOS leaf cells. Section 7.2 then presents an overview of the general algorithm, and the final three sections describe specific methods for synthesizing selected layout objects.

7.1. BOUNDARY DEFINITIONS FOR CMOS LAYOUT

The question of the number and nature of boundary definitions, required to generate a complete CMOS layout, is an important one. A first inspection of the problem suggests that one boundary per mask layer is both necessary and sufficient for layout synthesis. Indeed, the necessity of one boundary per mask layer is self-evident. However, it is possible for a single layer to play different roles in the layout according to how it interacts with other configurations present. For example, the diffusion layer should be treated differently when involved as channel, source/drain region or interconnect. A similar distinction must be made for the polysilicon layer, between transistor gates and polysilicon interconnect. Consequently, a thorough analysis of all the possible roles of each mask layer is required, in order to determine a complete set of boundaries.

7.1.1. Boundaries for the Polysilicon Layer

As was briefly discussed above, it is often necessary to distinguish between transistor gates and polysilicon wires during the layout synthesis process. Figure 7.1 illustrates the need for multiple boundaries on the polysilicon layer. Part (a) of the

LEGEND

- Polysilicon layer
- Diffusion layer

Figure 7.1. Boundary definitions for the polysilicon layer

figure shows a single transistor which is part of the layout, along with a second transistor we wish to merge with the existing design. If these two transistors are of the same size, then the boundary of Figure 7.1 (a) determines how far to the left the new transistor gate may be moved without causing a violation. If however, we wish to determine the left-most position of an unrelated polysilicon wire, then a new boundary drawn as in Figure 7.1 (b) is required. Hence, two distinct boundaries must be used for the polysilicon layer. These two boundaries are referred to as the "gate boundary" and "poly boundary" as shown in Figure 7.1 (a) and Figure 7.1 (b), respectively.

7.1.2. Boundaries for the Diffusion Layer

A similar discussion is applicable to the diffusion layer. Indeed it is necessary to distinguish between two unrelated diffusion regions at different potentials, and the case of two diffusion rectangles that are to be electrically connected. Figure 7.2 (a) illustrates the boundary necessary to restrict the movement of an unrelated diffusion region, such as a diffusion contact carrying a different electrical signal. The required boundary should be exactly one diffusion-diffusion spacing away from the active device. Figure 7.2 (b) shows the boundary required in the case of related diffusions, such as two transistors connected in series. The flexible boundary that restricts the movement of the new diffusion region should now be drawn through the existing device at a distance of one poly-diffusion spacing away from the transistor gate. Subsequent sections refer to these boundaries as the "diffusion boundary" and the "active boundary", respectively.

(a) Boundary for positioning unrelated diffusions

(b) Boundary for merging related diffusion regions

LEGEND

Polysilicon layer
Diffusion layer
Contact Window

Figure 7.2. Boundary definitions for the diffusion layer

7.1.3. Boundaries for the Metal Layer

The metal layer presents a slightly more complicated situation. As was the case for the polysilicon and diffusion layers, it is necessary to distinguish between unrelated metal objects and those at a common potential. However, the layout style used by the system allows metal wires to be implemented in both the horizontal and vertical directions. Hence, we need to avoid a situation where two metal polygons that are electrically connected cause a metal to metal spacing violation. Such would be the case, for example, of a very short horizontal metal wire with a contact window at both ends. Special measures must be taken to ensure that the metal cover for the contact window of the right end of the wire is positioned correctly with respect to the metal cover for the left end.

In spite of the increased complexity, two boundaries are still sufficient to ensure correct designs. A first boundary serves to restrict the position of metal objects carrying a different electrical signal. It is shown as a dashed line in Figure 7.3 (a). This boundary is referred to as the "metal boundary" in subsequent sections. The second boundary defined for the metal layer ensures that metal covers are appropriately spaced, regardless of electrical potentials. This boundary is called "cover boundary" and is illustrated in Figure 7.3 (b). If a new metal cover is to be added to the layout which carries the same signal as an existing vertical wire, it is convenient to allow this new metal cover unrestricted movement with respect to the metal wire while insuring sufficient spacing between the metal covers. That purpose is served by the "cover boundary".

(a) Purpose of the Metal Boundary

(b) Purpose of Cover Boundary

LEGEND

Figure 7.3. Boundary definitions for the metal layer

7.1.4. Other Boundaries

The remaining mask layers follow the rule of one boundary per layer. For a single metal process, these additional layers include contact windows and p-plus and n-plus implants. For processes which include a second metal layer, a single boundary is sufficient since Picasso restricts the use of such a layer to i/o purposes. Wells are not handled directly by the layout synthesis algorithm, since they can easily be added to the layout as a post-processing step.

7.2. OUTLINE OF THE CELL LAYOUT ALGORITHM

Picasso builds layout geometries by performing a one-dimensional compaction of the symbolic representation along the horizontal direction. The absolute vertical position of each object is determined by the cell structure. Flexible boundaries are used to represent physical constraints and serve as a guide in the determination of the position and shape of layout objects.

7.2.1. Formation of Symbolic Layout Slices

The first step of the layout synthesis algorithm is the partitioning of the symbolic layout into "slices". A slice is a data structure comprised of all symbolic layout primitives at a common vertical virtual grid point. A horizontal metal wire is included in the slice which contains its right-most end point.

7.2.2. Generation of Layout Geometries

Following the grouping of symbolic objects into slices, detailed layout geometries are generated for individual slices. The horizontal compaction algorithm operates from left to right by processing each slice in turn. Following the generation of layout geometries for all the objects and wires present in a slice definition, flexible boundaries are created to restrict the geometries generated for the following layout slice. The order in which objects are processed within each slice is important. Objects on the lines of diffusion, such as transistors and diffusion contacts are handled first. The reason is that transistor gates, for example, serve as reference points for potential polysilicon wires present in the same slice. Polysilicon objects, such as wires, poly-contacts and I/O ports are generated next. Finally, all metal wires for which both endpoints have been positioned, as a result of generating new contact windows, are included in the layout.

7.2.3. Representation of Constraints

Flexible boundaries essentially serve two purposes, as illustrated in Figure 7.4. First, they are used as a guide for the generation of objects in the current slice. In addition, they serve to represent physical constraints imposed on objects in subsequent slices. Clearly, this dual purpose cannot be served by a single structure, if constraints are to be built incrementally, as the layout is generated. Hence, Picasso maintains two versions for each boundary defined in Section 7.1. One version is termed "current" and is used to represent constraints that should be used for layout synthesis. The other version is referred to as the "next" boundary, and represents constraints on objects contained in the next slice. Following the treatment of all the objects present in the current slice, the "next" boundaries become "current", and the next slice becomes the current slice.

7.2.4. Summary of Layout Synthesis Algorithm

The general algorithm used for generating final mask geometries can be summarized as follows:

- Form slices from symbolic layout
- For each slice do:
 - Process diffusion objects (diffusion contacts, and transistors)
 - Process polysilicon objects (ports, contacts and wires)
 - Process all metal interconnect
 - Map "next" boundaries onto "current" boundaries
- Draw power busses, wells and complete implant masks

Figure 7.4 Dual purpose of flexible boundaries

7.3. CREATING LINES OF DIFFUSION

Generating chains of transistors is the most important task of CMOS layout synthesis. It is also the most difficult one since it involves a large number of potentially conflicting considerations. In terms of design rules, minimum dimensions of polysilicon and diffusion must be respected as well as poly overlap and diffusion overlap of the active region. From the topological point of view, transistors must be laid out in such a way as to make it possible to chain many devices in a common diffusion region, and also allow polysilicon wires extending from transistor gates to be completed. The task is further complicated by the presence of two distinct boundaries associated with each layer. Different layout problems will require the use of a different combination of boundaries, or will modify the interpretation given to individual boundaries.

7.3.1. Generation of Wide Transistors

Wide transistors, in the Picasso terminology, are transistors for which $W > L$. They are the most common layout objects encountered for most designs. Figure 7.5 illustrates the use of existing flexible boundaries in the determination of a transistor's shape. Figure 7.5 (a) shows an existing transistor with its associated "active", "gate" and "poly" boundaries, as well as the initial outline of a new transistor to be added. The vertical range of the active region of the existing transistor is referred to as the "active range". The steps involved in generating mask geometries for the new device can be summarized as follows:

- Determine position of left edge of diffusion rectangle - This is achieved by considering the position of the active boundary, as shown in Figure 7.5 (a).
- Prepare gate boundary at top and bottom - Determine the position of the gate at both the top and bottom of new transistor range. If inside the active range, align gate with poly boundary. If in line with active range, align gate with gate boundary. If outside active range, adjust gate position away from active edge. See Figure 7.5 (b).
- Ensure Space for Gate Extensions - Make further adjustments to the gate boundary to ensure that sufficient space exists at the top and bottom of the active region for gate extensions to be drawn.
- Filter gate boundary - Apply filter function to the gate boundary to remove unnecessary geometries and to ensure that the diffusion overlap of active region is sufficient at both ends.
- Generate Transistor Gate and Extensions - Draw polygons on the polysilicon layer to represent the gate itself and its extensions at the top and bottom of the active region.
- Compute position of right edge of diffusion rectangle - Ensure sufficient diffusion overlap of active region, as shown in Figure 7.5 (c).
- Generate Diffusion Rectangle - Draw a single polygon on the diffusion layer to represent the transistor's diffusion region.
- Adjust Boundaries - Make necessary changes to boundary positions in preparation for future layout objects.

(a) A sample transistor design problem

(b) Gate boundary position following diffusion left edge

(c) Final geometries generated

LEGEND

Polysilicon layer

Diffusion layer

Figure 7.5 Generation of wide transistors

7.3.2. Generation of Weak Transistors

Weak transistors are transistors whose aspect ratio are such that $W < L$. In other words, the gate is longer than it is wide. Their treatment is similar to wide transistors, but is simplified in two ways. First, since weak transistors are relatively few, in typical designs, it is feasible to restrict their geometry without ensuing a significant area penalty. Furthermore, it is very unlikely that multiple weak transistors will have to be chained together. Hence, efficient methods for chaining such devices are not critical. Figure 7.6 summarizes the geometries supported by Picasso for weak transistors.

Figure 7.6 Weak transistors supported by Picasso

7.3.3. Generation of Diffusion Contacts

Diffusion contacts serve as an interface between the diffusion lines and metal layer. They allow for connections to power rails, substrate contacts as well as the completion of some signal nets with metal connections. Figure 7.7 summarizes the types of diffusion contacts supported by the current implementation of the Picasso system. The structure shown in Figure 7.7 (a) is the most common occurrence and is introduced in diffusion chains wherever a net is to be completed with a metal connection. The algorithm used to determine the position of such an object is similar to that used to determine the position of the left edge of a wide transistor's diffusion rectangle.

Figure 7.7 (b), on the other hand, illustrates a split contact structure used for connections to a power rail. In order to increase immunity against latch up, split contacts are preferred to implement connections to Vdd or Vss. This produces a substrate contact for each transistor diffusion connection to a supply rail.

Figure 7.7 Diffusion contact structures

7.3.4. Generation of Diffusion Bridges

As was mentioned in an earlier section, the horizontal position of layout objects is determined by parameters of the cell structure. It is therefore possible for the diffusion regions of two elements of a diffusion chain to be separated by a gap in the vertical direction, as illustrated in Figure 7.8. A diffusion bridge is a vertical wire that serves to fill the gap between two such elements. It is added to the layout only after both layout objects to be linked have been completed. However, care must be taken to ensure that such a wire is feasible, when determining the position of diffusion objects involved.

Figure 7.8 Generation of diffusion bridges

7.4. GENERATING GEOMETRIES FOR POLYSILICON OBJECTS

7.4.1. Generation of Polysilicon Contacts

In order to make use of the smallest possible geometries allowed by design rules, the Picasso framework makes it possible to define a number of polysilicon contact structures. Figure 7.9 (a) illustrates the two structures which are currently supported. The first one is a simple square structure, which suffices in most cases. The second structure, however, is geared towards processes which apply a different overlap rule in the direction of an incident metal wire.

The main purpose of polysilicon contacts is to allow a connection between a metal wire and a transistor gate. Hence, the primary consideration in designing such objects is to ensure that a polysilicon wire is feasible between the contact window and the target transistor. Special care must be taken in determining the position of a polysilicon contact, to avoid "burying" it in between two unrelated objects that may preclude the implementation of polysilicon wires emerging from it. Figure 7.9 (b) contains an example of this undesirable situation, which can be avoided by filtering the polysilicon boundary before computing the left-most position for the object. The scenario depicted in Figure 7.9 (c), where a discontinuity exists between a contact window and the intended polysilicon wire, is also highly undesirable. It can be avoided by constraining the polysilicon boundary, following the implementation of the contact window, by adding the appropriate segment to the boundary definition.

7.4.2. Generation of Polysilicon Wires and Ports

Whereas the preferred direction of implementation for diffusion chains is horizontal, polysilicon wires are predominantly vertical. They serve to interconnect transistors and diffusion contacts, contained in the same slice definition, and may be extended to

(a) Polysilicon contact structures supported

(b) Contact positioned too far to the left

(c) Discontinuity between contact and wire

LEGEND

Figure 7.9 Generation of polysilicon contacts

the top or bottom edge of the cell, in order to implement an I/O port. Such polysilicon ports may be situated at the left-most position allowed by the polysilicon boundary, or alternatively can be "snapped" to the nearest I/O grid location.

7.5. GENERATING GEOMETRIES FOR METAL WIRES

Metal wires are the last objects generated for each slice definition. Vertical wires are synthesized first, using the metal boundary as a guide. A boundary filter is applied, in order to avoid strange geometries. The filter function used for vertical metal wires is similar to other boundary filters discussed so far, with the exception that the x-coordinates of the end points of the wire may not be known before hand. An additional constraint is therefore imposed on the filtering problem, so as to limit the horizontal freedom of vertical wires. Finally, horizontal wires whose right end points reside in the current slice are included in the layout. Because there is no vertical compaction phase in the generation of fixed-pitch cells, all horizontal wires are implemented as straight wires. The vertical position is determined by the cell structure, described in Chapter 3, and the coordinates of the end points are determined by the position of contact windows and/or vertical metal wires.

7.6. SUMMARY

Picasso's layout synthesis algorithm relies on a partitioning of the symbolic layout into "slices". Flexible boundaries are used to represent physical constraints and serve as a guide in the determination of the position and shape of layout objects.

CHAPTER 8

EXPERIMENTAL RESULTS

This chapter presents some experimental results produced by Picasso. It illustrates the behavior of the system on two specific synthesis examples, which reflect the typical complexity of the population of leaf cells, targeted by this research effort. Two different cell structures are used, to illustrate the adaptability of the system to the physical design environment. The last section of the chapter explores the impact of qualitative changes of some process design rules.

8.1. JK FLIP-FLOP EXAMPLE

The first example cell is a JK flip-flop, shown in Figure 8.1. This cell exhibits a number of features that make it difficult to realize. The logic description is a mixture of random logic and transmission gates that implement the front end and latch stages respectively. There are two long feed-back nets (Q and Q-) that complicate the routing phase. Finally, there are two feed-back gates (one for each latch stage) implemented with weak transistors.

For the first example, this flip-flop was generated using a cell structure which allows a maximum of two metal lines in each routing region, a restriction which is common in hand designs. Because of this very limited space for the generation of internal metal wires, and the presence of a large number of metal nets in the circuit description, the first two passes of the placement algorithm have failed to produce a routable solution. Each failed attempt has resulted in a net segment being removed from the net table and assigned as external to the cell. Consequently, the final layout geometries produced by the system require an external connection for segments of both feed-back nets Q and Q-.

Figure 8.1. JK flip-flop logic schematic.

8.1.1. Transistor Level Design

Recall that transistor level design is based on the definition of logic clusters. For the JK flip-flop example, seven clusters were identified by the system, based on the gate patterns summarized in Appendix B. The seven clusters are highlighted using dashed lines in Figure 8.1. Figure 8.2 summarizes the transistor structures generated for each cluster. The structure generated for C-1 is based on a special AND-OR-INVERT pattern which recognizes the presence of complementary inputs. C-2 and C-3 also correspond to predefined structures for back to back inverters, and back to back NAND-INVERT, respectively. A polysilicon feed through is added to the basic structure for C-2, since signal Q^- is a cell level output. Finally, clusters C-4, C-5, C-6 and C-7 correspond to simple gates and transmission gates.

Figure 8.2. Transistor level structures generated for JK flip-flop.

8.1.2. Cluster Placement

Cluster placement is based on an affinity function defined on clusters, which considers both morphology and connectivity. Figure 8.3 illustrates the data structures used to reflect these two criteria. Figure 8.3 (a) shows the connectivity graph which was extracted from the cluster definitions for the given set of parameters. The weights assigned to type-1, type-2 and type-3 connections were 4, 2 and 1, respectively. Figure 8.3 (b), on the other hand, contains the tile representation for each cluster.

A trace of the placement algorithm is shown in Figure 8.4. The first cluster to be placed is C-5 since it is the cluster with the highest affinity to others with 30.5. The remaining six clusters are placed by first selecting the cluster with highest affinity to placed clusters and then evaluating the merit of the eight possible placements. The merit of placement involving mirroring of a placed cluster is only evaluated if that cluster doesn't form a diffusion chain with another placed cluster. Figure 8.5 shows the detailed transistor placement obtained. It is comprised of forty slices, labeled 0-39.

(a) Connectivity graph

(b) Tile representation for clusters

Figure 8.3. Data structures used for cluster placement operation.

```

; Wc = 1.0
; Wm = 1.0
; a = 0.25
; W1 = 4
; W2 = 2
; W3 = 1
;
; Computing Affinity: (C_1 19.5)
; Computing Affinity: (C_2 19.75)
; Computing Affinity: (C_3 27.0)
; Computing Affinity: (C_4 27.5)
; Computing Affinity: (C_5 30.5)
; Computing Affinity: (C_6 21.75)
; Computing Affinity: (C_7 12.0)
; Placing: (C_5 +)
; Computing Affinity to Placed Clusters: (C_1 1.5)
; Computing Affinity to Placed Clusters: (C_2 6.5)
; Computing Affinity to Placed Clusters: (C_3 7.75)
; Computing Affinity to Placed Clusters: (C_4 10.75)
; Computing Affinity to Placed Clusters: (C_6 4.0)
; Computing Affinity to Placed Clusters: (C_7 0.0)
; Evaluating Merit: (C_4 FIRST PLUS 0.0)
; Evaluating Merit: (C_4 FIRST MINUS 0.0)
; Evaluating Merit: (C_4 LAST PLUS 2.0)
; Evaluating Merit: (C_4 LAST MINUS 1.0)
; Placing: (C_4 LAST +)
; Computing Affinity to Placed Clusters: (C_1 3.5)
; Computing Affinity to Placed Clusters: (C_2 14.5)
; Computing Affinity to Placed Clusters: (C_3 10.75)
; Computing Affinity to Placed Clusters: (C_6 6.75)
; Computing Affinity to Placed Clusters: (C_7 1.0)
; Evaluating Merit: (C_2 FIRST PLUS -8.0)
; Evaluating Merit: (C_2 FIRST MINUS -8.0)
; Evaluating Merit: (C_2 LAST PLUS -6.5)
; Evaluating Merit: (C_2 LAST MINUS -6.5)
; Placing: (C_2 LAST +)
; Computing Affinity to Placed Clusters: (C_1 1.5)
; Computing Affinity to Placed Clusters: (C_3 14.5)
; Computing Affinity to Placed Clusters: (C_6 8.25)
; Computing Affinity to Placed Clusters: (C_7 0.0)
; Evaluating Merit: (C_3 FIRST PLUS -9.5)
; Evaluating Merit: (C_3 FIRST MINUS -10.5)
; Evaluating Merit: (C_3 LAST PLUS -16.5)
; Evaluating Merit: (C_3 LAST MINUS -16.5)
; Evaluating Merit: (C_3 LAST PLUS FLIP -16.5)
; Evaluating Merit: (C_3 LAST MINUS FLIP -16.5)
; Placing: (C_3 FIRST +)
; Computing Affinity to Placed Clusters: (C_1 3.5)
; Computing Affinity to Placed Clusters: (C_6 15.75)
; Computing Affinity to Placed Clusters: (C_7 0.0)
; Evaluating Merit: (C_6 FIRST PLUS -12.0)
; Evaluating Merit: (C_6 FIRST MINUS -14.0)
; Evaluating Merit: (C_6 LAST PLUS -27.25)
; Evaluating Merit: (C_6 LAST MINUS -27.25)
; Evaluating Merit: (C_6 LAST PLUS FLIP -27.25)
; Evaluating Merit: (C_6 LAST MINUS FLIP -27.25)
; Placing: (C_6 FIRST +)
; Computing Affinity to Placed Clusters: (C_1 7.5)
; Computing Affinity to Placed Clusters: (C_7 2.0)
; Evaluating Merit: (C_1 FIRST PLUS -5.0)
; Evaluating Merit: (C_1 FIRST MINUS -5.0)
; Evaluating Merit: (C_1 LAST PLUS -25.0)
; Evaluating Merit: (C_1 LAST MINUS -25.0)
; Evaluating Merit: (C_1 LAST PLUS FLIP -25.0)
; Evaluating Merit: (C_1 LAST MINUS FLIP -25.0)
; Placing: (C_1 FIRST +)
; Computing Affinity to Placed Clusters: (C_7 8.0)
; Evaluating Merit: (C_7 FIRST PLUS -2.0)
; Evaluating Merit: (C_7 FIRST MINUS 0.0)
; Evaluating Merit: (C_7 LAST PLUS -28.0)
; Evaluating Merit: (C_7 LAST MINUS -28.0)
; Evaluating Merit: (C_7 FIRST PLUS FLIP -2.0)
; Evaluating Merit: (C_7 FIRST MINUS FLIP 0.0)
; Evaluating Merit: (C_7 LAST PLUS FLIP -28.0)
; Evaluating Merit: (C_7 LAST MINUS FLIP -28.0)
; Placing: (C_7 FIRST -)
(C_7 C_1 C_6 C_3 C_5 C_4 C_2)
> [dribble]

```

Figure 8.4. Trace of placement algorithm for JK flip-flop.

Figure 8.5. Detailed transistor placement for JK flip-flop.

8.1.3. Routing of Internal Nets

Following cluster placement, adjacent clusters with compatible diffusion signals can be merged, by removing a number of diffusion contacts from the symbolic representation. Also, each signal net which has not been completely implemented via a common diffusion connection or a vertical polysilicon wire should be extracted from the symbolic layout and included in a net table. Both the updated symbolic representation and net table are supplied as inputs for the routing algorithm. For the JK flip-flop example, the number of slices in the symbolic representation has been reduced from 40 to 36 and a net table comprised of 10 metal nets has been created. These two structures are shown in Figure 8.6.

The result of the routing phase is shown in Figure 8.7. All the net segments present in the net table have been instantiated as a collection of two point segments on the virtual routing grid. Furthermore, following the routing operation, a number of polysilicon wires have become obsolete and hence have been removed from the symbolic layout representation. For example, a visible polysilicon wire for signal CK extends all the way to the n-region, in slice number 12 of Figure 8.6, in order to provide maximum freedom for the routing algorithm to implement a connection to that gate signal at any point within the slice. However, since the routing algorithm has implemented that connection in the p-region, the wire extending to the n-region has become obsolete and can be removed from the database. The removal of two such polysilicon wires has resulted in a further reduction in the number of slices from 36 to 34.

Figure 8.6. Structures provided as inputs to the routing algorithm.

Figure 8.7. Detailed symbolic representation for JK flip-flop.

8.1.4. Layout Synthesis

Picasso builds layout geometries by performing a one-dimensional compaction of the symbolic representation along the horizontal direction. Following the generation of layout geometries for all the objects and wires present in a slice definition, flexible boundaries are created to restrict the geometries generated for the following slice. Figure 8.8 illustrates the behavior of the layout synthesis algorithm by showing the partial solution produced following the generation of the first few slices.

Figure 8.9 illustrates the final layout produced for the complete JK flip-flop example. Two external connections are required in order to complete nets Q and Q-, respectively. For that reason two I/O ports exist for each net.

Figure 8.8. Partial layouts after first few slices.

Figure 8.9. Complete layout generated for JK flip-flop example.

8.2. ALTERNATIVE CELL STRUCTURE

This section explores the impact of the cell structure definition on the synthesis process, in order to demonstrate Picasso's adaptability to different physical environments. The JK flip-flop example was repeated with a new cell structure definition which now allows three metal tracks to be defined within each active region. Given the new cell structure, the topological expansion phase succeeded the first time. Figure 8.10 illustrates the new symbolic layout generated for the cell which is now comprised of 33 slices. Figure 8.11 contains the detailed layout produced under the new cell structure. External connections are no longer required to complete internal nets.

Figure 8.11 Detailed layout produced for JK flip-flop under new cell structure

8.3. LOGIC SYNTHESIS EXAMPLE

The second example will serve to illustrate Picasso's performance on CMOS complex gates. The example cell is a full adder comprised of two complex gates C-1 and C-2, each corresponding to one cluster, as shown in Figure 8.12. Whereas transistor level design for the JK flip-flop example relied on a number of pre-defined transistor structures, the graph-theoretical method for complex gates must be invoked for each cluster of the full adder example. As was mentioned in Chapter 4, the complex gate compiler is able to accept constraints on the number of internal metal tracks available within each routing region. Figure 8.13 shows a first cell outline generated by Picasso given a cell structure which imposes no upper bound on the number of metal tracks used within the cell. The emphasis is put on optimal chaining of transistors alone. If, however, an upper bound of two is imposed on the number of tracks for each region, the system produces the outline shown in Figure 8.14. This new solution is not optimal in terms of transistor chaining but satisfies the constraint of two metal tracks per routing region. Figure 8.15 presents the symbolic representation for the latter case, whereas Figure 8.16 contains the detailed layout generated.

Figure 8.12. Full adder logic description.

Figure 8.13. First outline generated for full adder.

Figure 8.14. Second outline generated for full adder.

Figure 8.15. Symbolic layout generated for full adder example.

Figure 8.16. Detailed layout generated for full adder example.

8.4. IMPACT OF QUALITATIVE PROCESS DESCRIPTION

The purpose of this section is to illustrate how the Picasso system can adapt to qualitative differences between processes. The qualitative process description affects the system in two ways. First, the definition of the interaction between the layers determines the nature of the flexible boundaries that will be set by each layout object. For example, if there exists a constraint on the relative position of n+ and p+ implant masks with respect to polysilicon wires, then all objects on the polysilicon layer will affect both implant boundaries, and vice versa. However, if the target process is making use of silicided polysilicon, then the polysilicon-implant constraint can be removed.

The second impact of the qualitative process description on the system is in the generation of layout geometries for each object. An example of this situation was already given in Section 7.4, during the discussion of the different polysilicon contact structures supported by Picasso. The shape of wires on the polysilicon and metal layers can also be constrained in a similar fashion. Assume for example, that the target process prohibits the use of non-Manhattan geometries, in order to ensure compatibility with other CAD tools. Then, Picasso would automatically make use of a slightly different set of methods for layout generation.

In order to illustrate how the Picasso system can adapt to such qualitative differences, the full adder example was repeated using only Manhattan geometries. The result is shown in Figure 8.17. Note that even though the final layout is rectilinear, the flexible boundaries are still represented internally using forty-five degree bends where applicable. Hence, the new layout is only one per cent larger than the original one.

Figure 8.17. Detailed layout for full adder with Manhattan geometries only.

8.5. COMPLEXITY ISSUES

This section addresses the time complexity of the algorithms used in the Picasso system. Complexity is defined as the rate of growth of the amount of a given resource required, with respect to the size of the problem. For leaf cell synthesis, it is customary to use the number of transistors in the circuit as a measure of the size of the problem. Furthermore, it is normally assumed that the number of transistors, logic gates, and internal nets, are all proportional to one another.

Transistor level design is performed in linear time with respect to problem size. The cluster formation process considers each gate in turn and attempts to include it into an existing cluster. The creation of stages is performed in constant time since at most four stages may be created. The same reasoning applies to the ordering of stages (sub-graphs) with the graph. The generation of a graph traversal for each stage is done in linear time since it only depends on the type and the size of each stage. Thus, transistor level design is $O(n)$ in the worst case.

The time required for cluster placement grows as the square of the number of clusters, which is assumed to be proportional to the problem size. Indeed, for the i th unplaced cluster, a comparison with $(n - i - 1)$ other clusters is required. Similarly, the routing algorithm is quadratic. The assignment of net segments to cell regions requires the comparison of a given segment with all the other unassigned segments, in order to select the next segment to be processed. Hence, each pass of the topological expansion is $O(n^2)$ in the worst case, even though the number of clusters and the number of nets is much less than the number of transistors on average. Because Picasso will assign nets as external upon failure of the routing process, both placement and routing steps may be repeated as often as there are nets in the circuit description. Thus, the complete topological expansion process is $O(n^3)$ worst case. In practical terms however, internal signal nets are sufficiently local in nature to ensure that the number of external tracks

is bound by a small constant on average, and hence, it is reasonable to say that the algorithm is $O(n^2)$ on average.

Finally the layout synthesis process considers each object in turn and only applies local constraints to determine geometries. Its complexity is therefore $O(n)$. However, for most of the circuits produced by Picasso thus far, layout synthesis accounts for more than 90% of the execution time. This is due to the high overhead of creating and maintaining flexible boundaries throughout the synthesis process. In spite of its linear time complexity, the layout synthesis module is likely to remain the dominant contributor to execution time for the range of cells targeted by the Picasso system.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

Leaf cell synthesis is now a mature technology which is awaiting general acceptance by the integrated circuit design industry. The cell generation problem as well as its potential benefits have become well understood, in recent years. This has resulted in an ever growing research effort, spread over both the academic and industrial scenes. The purpose of this chapter is to establish the contribution of the Picasso system to the state-of-the-art in leaf cell synthesis, and make recommendations for future research.

9.1. CONTRIBUTIONS

Picasso is a very broad system which touches upon many different areas. It presents a number of enhancements over existing approaches which range from system level concepts to performance optimization of individual synthesis steps.

Picasso is one of the first cell synthesis systems to consider the important problem of mapping logic-level descriptions into transistor structures. This feature allows the same logic circuit to be realized by a different transistor structure when subjected to different constraints. Furthermore, the system generates transistor netlists which are amenable to efficient realizations in the target layout style. Transistor level design is based on a combination of predefined structures for simple clusters, to insure maximum efficiency, with an extensive graph-theoretical framework for complex gates. Picasso's synthesis method for complex gates imposes fewer restrictions than other approaches on the nature of layouts produced, and is capable of minimizing the amount of internal interconnect, in addition to forming optimal chains of transistors.

Some of the salient features of the topological expansion mechanism include the ability to limit the height of the cell by allocating external tracks. The formation of logic clusters allows the introduction of a placement hierarchy which reduces the search space. A composite metric is used to guide the cluster placement algorithm which applies connectivity and optimal chaining considerations simultaneously. Although routing over transistor chains is not new, the routing algorithm makes better use of residual routing areas resulting from unequal transistor sizes.

Detailed layout synthesis is the forte of the Picasso system. It introduces a new compaction mechanism based on flexible boundaries, which is optimal in one dimension. Although the concept of forbidden regions has been applied previously to the river routing problem, on a single layer, the extension of that concept for the generation of a complete CMOS layout is new. The final shape of layout objects is determined during the compaction phase, using boundary definitions as a guide, which eliminates the need for the explicit insertion of jogs in the symbolic layout. Furthermore, non-Mahattan geometries, such as forty-five degree bends in transistor gates and wires, are handled naturally by flexible boundaries. Although such geometries do not generally yield a significant reduction in cell area, they can be used to improve the predictability of transistor behavior as well as reduce parasitic capacitances.

Finally, the use of an object-oriented programming paradigm allows the system to easily support a more advanced form of process independence. Indeed, the Picasso system is capable of accommodating certain kinds of qualitative process variations, in addition to quantitative changes in physical design rules.

9.2. SYSTEM LIMITATIONS

The following system limitations should be addressed in the short term, in order to enhance the functionality of the current prototype.

First, in an effort to support a true top-down design methodology, it is necessary for a cell compiler to be able to accept hard constraints on the position of I/O ports. Whereas Picasso strives to follow a preferred ordering for such connections, wherever possible, the system should be modified to always accommodate I/O constraints when requested by the user. Because Picasso uses a hierarchical placement strategy, changes would be required in both the transistor level design and cluster placement modules.

From a system perspective, the ability to support other design styles would be an asset. For example, dynamic circuits often result in an unequal number of p-type and n-type devices. A new cell structure is required in order to efficiently support such circuits. Also, large circuits with uniform transistor sizes may be better served by a cell topology comprised of two lines of diffusion in each region, instead of one. Finally, an overconstrained layout style such as gate array or sea-of-gates array could be achieved at a minimal cost, and would increase the flexibility of the system.

On the technology front, Picasso is begging for a second level of metalization. Traditionally, the metal2 layer has been confined to a very limited role for leaf cell design. This was mainly due to the excessive area consumption resulting from the implementation of vias inside the cell area. Recent advances in process technology, however, promise to make such vias feasible in the near future. The addition of a second metal layer will impact in two ways on the Picasso system. First, additional boundaries must be established to support the new layer, depending on its target utilization. This is a relatively simple task. The impact of the second metal layer on the cell topology poses a much more difficult problem. Indeed, an additional synthesis step

is required, in order to assign I/O signals to either the polysilicon layer or the metal2 layer. Furthermore, the routing algorithm will have to be modified significantly, in order to make efficient use of the additional resource.

9.3. FUTURE WORK

In addition to the proposed enhancements to the Picasso system, the results presented in this thesis can serve as the basis for other challenging work in related areas. In particular, Picasso could be used as the corner stone for the implementation of the design flow presented in Figure 1.1. Such a system would encompass both electrical and physical optimization modules, that would allow it to fully explore the structural domain (modifications to the logic netlist) as well as the physical domain. Extensions to the cell compiler would also allow the exploitation of selected bipolar devices within the CMOS context.

The Picasso system could also be used as part of an automatic system for customizing process design rules. Currently, the determination of physical and electrical design rules for a new process is considered by most to be witchcraft, because of the inherent difficulty in obtaining feedback for any particular design decision. A cell compiler could be used to automatically generate a large number of complex test structures which would provide a thorough exploration of the design space.

Other interesting applications include the use of flexible boundaries for two-dimensional compaction. This could be achieved either by extending the role of existing boundaries or by introducing a complete new set of boundaries for the vertical direction. The flexible boundary concept is also amenable to perform compaction of curvilinear geometric structures, a domain outside the reach of constraints-based compaction systems.

APPENDIX A: GLOSSARY OF TERMS

- AOI:** And-or-invert.
- ASIC:** Application specific integrated circuit.
- BNF:** Backus-Naur form.
- CAD:** Computer-aided design.
- Cluster:** Group of logic gates formed to facilitate the generation of a transistor netlist as well as to introduce a placement hierarchy.
- Complex gate:** Combinatorial structure comprised of series-parallel combinations of MOS transistors.
- Euler path:** In graph theory; a path which traverses all the edges of a graph exactly once.
- Flavors:** Software package for object-oriented programming withing the Lisp framework.
- Flexible boundary:** Data structure used to represent physical constraints between layout objects.
- Gate matrix:** Layout style often used in leaf cell synthesis systems (see p.8).
- I/O:** Input/Output.
- Jog:** An abrupt change of direction in a wire.
- L:** Length of a transistor's active channel.

- Layer:** A collection of polygons associated with the same processing step.
- Net-table:** Data structure to represent the horizontal span of each net in a cell.
- PLA:** Programmable logic array.
- Routing grid:** Two-dimensional data structure to represent the topological ordering of metal segments for the implementation of internal nets.
- Slice:** A collection of layout objects at a common vertical grid position
- Vdd:** Positive power rail (normally 5V).
- VLSI:** Very large scale integrated (circuits).
- Vss:** Negative power rail (normally 0V).
- W:** Width of a transistor's active channel.
- Weak transistor:** A transistor whose aspect ratio is such that $W < L$.
- Wide transistor:** A transistor whose aspect ratio is such that $W \gg L$.
- Weinberger array:** Layout style for the implementation of large logic functions (see [Ullm]).

APPENDIX B: GATE PATTERNS FOR CLUSTER FORMATION

This appendix summarizes the gate patterns used during the cluster formation process in the current implementation of the Picasso system. Figure B.1 illustrates the gate patterns which are synthesized using the graph-theoretical framework presented in chapter 4. Figure B.2, on the other hand, contains the gate patterns for which predefined transistor level structures exist. Note that in both cases the order in which the patterns are stored in the database is important.

* Note: Indicates maximum number of inputs per gate.

Figure B.1. Summary of gate patterns for complex gates.

* Note: Indicates maximum number of inputs or additional inputs.

Figure B.2. Summary of gate patterns for pre-defined structures.

BIBLIOGRAPHY

- [BaAl]Baltus, D.G., Allen, J., "SOLO: A generator of efficient layouts from optimized MOS circuit schematics", in Proc. 25th Design Automation Conf., 1988, pp. 445-452.
- [BoWe]Boyer, D.G., Weste, N., "Virtual Grid Compaction Using The Most Recent Layers Algorithm", Proc. International Conference on Computer-Aided Design, 1985, pp.92-93.
- [ChKS]Cho, Y.E., Korenjak, A.J., Stockton, D.E., "FLOSS: An approach to automated layout for high-volume designs", in Proc. 14th Design Automation Conference, 1977, pp. 138-141.
- [ChHo]Chen, C.Y.R., Hou, C.Y., "A new layout optimization methodology for CMOS complex gates", in Proc. Int. Conf. on Computer-Aided Design, 1988, pp. 368-371.
- [Cho]Cho, Y.E., "A subjective review of compaction", in Proc. 22nd Design Automation Conference, 1985, pp. 396-404.
- [DBMA]Dholakia, S., Bush, S., Martinez, A., Asato, C., Walker, B., "Compilation of Standard Cell Libraries", Proc. IEEE 1986 Custom Integrated Circuits Conference, May 1986, pp. 343-346.
- [Deut]Deutsch, D.N., "Compacted Channel Routing", in Proc. International Conference on Computer-Aided Design, 1985, pp.223-225.
- [DGRS]Detjeus, E., Gannot, G., Rudell, R., Sangiovanni-Vincentelli, A., Wang, A., "Technology Mapping in MIS", Proc. International Conference on Computer-Aided Design, 1987, pp.116-119.
- [Dunl]Dunlop, A.E., "SLIM: The translation of symbolic layouts into mask data", in Proc. 17th Design Automation Conference, 1980, pp. 595-602.
- [GaTh]Gajski, D.D., Thomas, D.E., "Introduction to Silicon Compilation", in Silicon Compilation, Ed. by D. Gajski, Addison-Wesley Publishing Company Inc., 1988. pp. 1-48.
- [GlDo]Glasser, L.A., Dobberpuhl, D.W., "The Design and Analysis of VLSI Circuits", Addison-Wesley Publishing Company, 1985.
- [Hill]Hill, D.D., "Sc2: A Hybrid Automatic Layout System", Proc. International Conference On Computer-Aided Design, 1985, pp. 172-174.
- [Hsu]Hsu, C.P., "General River Routing Algorithm", Proc. 20th Design Automation Conf., 1983, pp.578-583.
- [KeKL]Keutzer, K., Kolwicz, K., Lega, M. "Impact of Library Size on the Quality of Automated Synthesis", Proc. International Conference on Computer-Aided Design, 1987, pp.120-123.

- [KiMS]Kim, J.H., McDermott, J., Siewiorek, D.P., "Exploiting Domain Knowledge in IC Cell Layout", IEEE Design and Test, August 1984, pp. 52-64.
- [KiGV]Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., "Optimization by Simulated Annealing", Science, Vol. 220, No. 4598, May 1983, pp.671-680.
- [KKLL]Kang, S.M., Krambeck, R.H., Law, H.F.S., Lopez, A.D., "Gate Matrix Layout of Random Control Logic in a 32-Bit CPU", Proc. 19th Design Automation Conference, 1982, 170-174.
- [KLPP]Kollarritsch, P.W., Lusky, S., Prasad, S., Potter, N., "CLAY: A Malleable-cell Multi-cell Transistor Matrix Approach for CMOS LAYout Synthesis", in Proc. Int. Conf. on Computer-Aided Design, 1988, pp. 142-145.
- [KoWe]Kollarritsch, P.W., Weste, N.H.E., "TOPOLOGIZER: An Expert System Translator of Transistor Connectivity to Symbolic Cell Layout", IEEE Journal of Solid-State Circuits, vol. SC-20, no. 3, June 1985, pp. 799-804.
- [KwKg]Kwon, Y.J., Kyung, C.M., "A fast heuristics for optimal CMOS functional cell layout generation", in Proc. Int. Symposium on Circuits and Systems, 1988, pp. 2423-2426.
- [LeCh1]Lefebvre, M.C., Chan, C.H., "Automatic Generation Of CMOS Standard Cell Layout from Logic Level Descriptions", Proc. 1986 Canadian Conf. on VLSI, October 1986, pp. 113-118.
- [LeCh2]Lefebvre, M.C., Chan, C.H., "Heuristics-based synthesis of CMOS complex gates", in Proc. 1988 Canadian Conf. on VLSI, October 1988, pp. 352-361.
- [LeCh3]Lefebvre, M.C., Chan, C.H., "Optimal ordering of gate signals in CMOS complex gates", in Proc. Int. Custom Integrated Circuits Conf., 1989.
- [LeCh4]Lefebvre, M.C., Chan, C.H., "Flexible Boundaries for Layout Synthesis", Proc. VLSI'89 International Conference, August 1989.
- [Li]Li, J.T., "Algorithms for Gate Matrix Layout", in Proc. Int. Symposium on Circuits and Systems, 1983, pp. 1013-1016.
- [Lin]Lin, S., "Computer Solutions of the Traveling Salesman Problem", Bell System Technical Journal, Vol. 44, No. 12, December 1965, pp.2245-2269.
- [LoLa]Lopez, A.D., Law, S.H., "A Dense Gate Matrix Layout Method for MOS VLSI", IEEE Journal of Solid-State Circuits SC-15(4), 1980, pp. 736-740.
- [MaDe]Mailhot, F., DeMicheli, G., "Automatic Layout and Optimization of Static CMOS Cells". Proc. International Conf. on Computer Design, 1988, pp.180-185.
- [MaHa]Maziasz, R.L., Hayes, J.P., "Layout Optimization of CMOS Functional Cells", Proc. 24th Design Automation Conference, 1987, pp. 544-551.

[MaSH]Marple, D., Smulders, M., Hegen, H., "An Efficient Compactor for 45 Degree Layout", Proc. 25th Design Automation Conference, 1988, pp.396-402.

[Most]Mosteller, R.C., "REST: A leaf cell synthesis system," in VLSI '81, John P. Gray, Ed., New York: Academic Press, 1981, pp. 163-172.

[NaBR]Nair, R., Bruss, A., Reif, J., "Linear Time Algorithms For Optimal CMOS Layout", in VLSI: Algorithms and Architectures, Bertollazi and Luccio Eds, North-Holland Pub., 1985, pp. 327-338.

[RoHB]Royal, N., Hunter, J., Buchanan, I., "A Case Study in Process Independence", Proc. 22nd Design Automation Conf., June 1985, pp. 591-596.

[RPVN]Royle, J., Palczewski, M., VerHeyen, H., Naccache, N., Soukup, J., "Geometrical Compaction in One Dimension for Channel Routing", Proc. 24th Design Automation Conf., 1987, pp.140-145.

[SKSA]Shiple, T., Kollaritsh, P., Smith, D., Allen, J., "Area Evaluation Metrics for Transistor Placement", in Proc. Int. Conf. on Computer Design, 1988, pp. 428-433.

[SSKT]Shiraishi, Y., Sakemi, J., Kutsuwada, M., Tsukizoe, A., Satoh, T., "A High Packing Density Module Generator for CMOS Logic Cells", Proc. 25th Design Automation Conf., 1988, pp.439-444.

[SSVS]Shin, H., Sangiovanni-Vincentelli, A.L., Sequin, C.H., "Two-Dimensional Compaction by Zone Refining", in Proc. 23rd Design Automation Conf., 1986, pp. 115-122.

[StNa]Stauffer, A., Nair, R., "Optimal CMOS Cell Transistor Placement: A Relaxation Approach", Proc. International Conference on Computer-Aided Design, 1988, pp.364-367.

[Sun]Sun, P.K., "An Octagonal Geometry Compactor", Proc. International Conf. on Computer Design, 1988, pp.190-193.

[UeVC]Uehara, T., vanCleemput, W.M., "Optimal Layout of CMOS Functional Arrays", IEEE Trans. on Computers, vol. c-30, no. 5, May 1981, pp. 305-312.

[Ullm]Ullman, J.D., "Computational Aspects of VLSI", Computer Science Press, Inc. Rockville, Maryland, 1984.

[VoES]Vopni, E.A., Elmasry, M.I., Sunter, S., "A Standard Cell Compiler Based on Gate Matrix Techniques", Proc. 1986 Canadian Conf. on VLSI, October 1986, pp. 101-106.

[WeEs]Weste, N.H.E., Eshraghian, K., "Principles of CMOS VLSI Design", Addison-Wesley Publishing Company, 1985.

[WiHW]Wing, O., Huang, S., Wang, R., "Gate Matrix Layout", IEEE Transactions on Computer-Aided Design CAD-4(3), July 1985, pp. 220-231.

[Wing]Wing, O., "Automated Gate Matrix Layout", Proc. Int. Conf. on Circuits and Systems, 1982, pp. 681-685.

[WiPF]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.

[WMND]Wolf, W.H., Mathews, R.G., Newkirk, J.A., Dutton, R.W., "Algorithms for optimizing two-dimensional symbolic layout compaction", IEEE Transactions on Computer-Aided Design, Vol. 7, No. 4, April 1988, pp. 451-466.

[XiKu]Xiong, X.M., Kuh, E.S., "Nutcracker: An Efficient and Intelligent Channel Spacer", in Proc. 24th Design Automation Conf., 1987, pp. 298-304.

[Yosh]Yoshimura, T., "An efficient channel router", in Proc. 21st Design Automation Conference, 1983, pp. 38-44.

END

1|6·0|2·90

FIN