National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file / Votre référence

Our file / Notre référence

NOTICE

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

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

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

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

AVIS

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

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

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

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

Canada

Module Generators for the Layout Synthesis of BiCMOS Input/Output Buffers

Kim-Minh Nguyen, B.Eng.

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

Department of Electronics, Carleton University
Ottawa, Ontario K1S 5B6
October 1993

© Copyright 1993, Kim-Minh Nguyen

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file *Votre référence*

Our file *Notre référence*

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

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

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

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

ISBN 0-315-89826-7

Canada

Name KIM-MINH NGUYEN

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Electronics Engineering

SUBJECT TERM

0544

SUBJECT CODE

U-M-I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0679
General 0289
Ancient 0290
Linguistics 0291
Modern
Literature 0401
General 0294
Classical 0295
Comparative 0297
Medieval 0298
Modern 0316
African 0591
American 0305
Asian 0352
Canadian (English) 0355
Canadian (French) 0593
English 0311
Germanic 0312
Latin American 0315
Middle Eastern 0313
Romance 0314
Slavic and East European

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion
General 0318
Biblical Studies 0321
Clergy 0319
History of 0320
Philosophy of 0322
Theology 0469

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
Archaeology 0326
Cultural 0327
Physical
Business Administration
General 0310
Accounting 0272
Banking 0770
Management 0454
Marketing 0338
Canadian Studies 0385
Economics 0501
General 0503
Agricultural 0505
Commerce-Business 0508
Finance 0509
History 0510
Labor 0511
Theory 0358
Folklore 0366
Geography 0351
Gerontology
History
General 0578

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science
General 0615
International Law and Relations 0616
Public Administration 0617
Recreation 0814
Social Work 0452
Sociology
General 0626
Criminology and Penology 0627
Demography 0938
Ethnic and Racial Studies 0631
Individual and Family Studies 0628
Industrial and Labor Relations 0629
Public and Social Welfare 0630
Social Structure and Development 0700
Theory and Methods 0344
Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
General 0285
Agronomy
Animal Culture and Nutrition 0475
Animal Pathology 0476
Food Science and Technology 0359
Forestry and Wildlife 0478
Plant Culture 0479
Plant Pathology 0480
Plant Physiology 0817
Range Management 0777
Wood Technology 0746
Biology
General 0306
Anatomy 0287
Biostatistics 0308
Botany 0309
Cell 0379
Ecology 0329
Entomology 0353
Genetics 0369
Limnology 0793
Microbiology 0410
Molecular 0307
Neuroscience 0317
Oceanography 0416
Physiology 0433
Radiation 0821
Veterinary Science 0778
Zoology 0472
Biophysics
General 0786
Medical 0760
EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences
General 0566
Audiology 0300
Chemotherapy 0992
Dentistry 0567
Education 0350
Hospital Management 0769
Human Development 0758
Immunology 0982
Medicine and Surgery 0564
Mental Health 0347
Nursing 0569
Nutrition 0570
Obstetrics and Gynecology 0380
Occupational Health and Therapy 0354
Ophthalmology 0381
Pathology 0571
Pharmacology 0419
Pharmacy 0572
Physical Therapy 0382
Public Health 0573
Radiology 0574
Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences
Chemistry
General 0485
Agricultural 0749
Analytical 0486
Biochemistry 0487
Inorganic 0488
Nuclear 0738
Organic 0490
Pharmaceutical 0491
Physical 0494
Polymer 0495
Radiation 0754
Mathematics 0405
Physics
General 0605
Acoustics 0986
Astronomy and Astrophysics 0606
Atmospheric Science 0608
Atomic 0748
Electronics and Electricity 0607
Elementary Particles and High Energy 0798
Fluid and Plasma 0759
Molecular 0609
Nuclear 0610
Optics 0752
Radiation 0756
Solid State 0611
Statistics 0463
Applied Sciences
Applied Mechanics 0346
Computer Science 0984

Engineering
General 0537
Aerospace 0538
Agricultural 0539
Automotive 0540
Biomedical 0541
Chemical 0542
Civil 0543
Electronics and Electrical 0544
Heat and Thermodynamics 0348
Hydraulic 0545
Industrial 0546
Marine 0547
Materials Science 0794
Mechanical 0548
Metallurgy 0743
Mining 0551
Nuclear 0552
Packaging 0549
Petroleum 0765
Sanitary and Municipal 0554
System Science 0790
Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Carleton University
Ottawa, Canada K1S 5J7

1912-1992

Thesis contains black & white
illustrations which when microfilmed
may lose their significance. The
hardcopy of the thesis is available
upon request from Carleton
University Library.

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

"Module Generators for the Layout Synthesis of BiCMOS Input/Output Buffers"

submitted by Kim-Minh Nguyen (B. Eng.) in partial fulfillment of the
requirements for the degree of Master of Engineering

A handwritten signature in cursive script, appearing to read "Martin", written over a horizontal line.

Thesis Supervisor

A handwritten signature in cursive script, appearing to read "J. S. Wright", written over a horizontal line.

Chairman, Department of Electronics

Carleton University

October 12, 1993

Abstract

This thesis presents a design automation methodology for the layout synthesis of standard input/output pad macros. The methodology hinges upon a collection of special purpose module generators for common building blocks such as transistors, resistors, diodes, etc. For each module generator, a shape function is developed which summarizes the physical dimensions of feasible layouts for that circuit element.

Issues related to the layout, floorplanning, routing, compaction, electrostatic discharge (ESD) and latch-up protection in the context of designing shape functions and module generators are discussed. Considerations made for the design issues are included in the prototype implementation. In addition, CMOS, TTL and ECL compatible buffer circuits are also presented. Example layouts synthesized by the module generators for a CMOS and BiCMOS technology are also presented.

Acknowledgments

I am thankful to my thesis supervisor, Dr. M.C. Lefebvre for all that he has done and been for me: as a friend, advisor, professor and fellow researcher. His continued interest and support throughout the course of my work was invaluable. I wish to extend my thanks to F. Black of Bell-Northern Research Ltd. for his feedback and advises on my work. Also, my sincere appreciation to my fellow graduate students and staff of the Department of Electronics for creating a superb work environment.

This research would not be possible without the financial support of Carleton University, the Natural Science and Engineering Research Council of Canada, and Micronet, a Network of Centres of Excellence focussing on microelectronics research and funded by the Government of Canada. In addition, I would like to acknowledge the Canadian Microelectronics Corporation for providing the workstations and software licenses used for the prototype implementation.

Finally, I wish to express my thanks to my family and friends for all their encouragement and undying confidence. This thesis is dedicated to my family.

Table of Contents

Abstract.....	i
Acknowledgments	ii
Table of Contents.....	iii
List of Figures.....	vi
Chapter 1 Introduction	1
1.1 Motivations	1
1.2 Objectives	1
1.3 Thesis Outline	2
Chapter 2 Design of Standard I/O Pad Macros	3
2.1 Typical Pad Configurations	3
2.1.1 Narrow Pad Configuration.....	3
2.1.2 Wide Pad Configuration.....	5
2.2 Latch-Up Prevention	6
2.2.1 Overview of Latch-Up.....	6
2.2.2 Layout Guidelines for Prevention Against Latch-Up	8
2.3 Electrostatic Discharge (ESD) Protection	9
2.3.1 ESD Protection Circuit	9
2.3.2 Layout Guidelines for ESD Protection	12
2.4 Mixed Levels Signal Compatibility.....	13
2.5 CMOS Compatible Buffer Circuits	14
2.5.1 CMOS-to-CMOS Output Buffer Circuit	14
2.5.2 CMOS-to-CMOS Input Buffer Circuit	15
2.5.3 CMOS-to-CMOS Bi-Directional Buffer Circuit	15
2.6 TTL Compatible Buffer Circuits	16
2.6.1 CMOS-to-TTL Output Buffer Circuit	16
2.6.2 TTL-to-CMOS Input Buffer Circuit	17
2.6.3 TTL Bi-Directional Buffer Circuit.....	18
2.7 ECL Compatible Buffer Circuits.....	18
2.7.1 ECL Output Buffer Circuit	18
2.7.2 ECL Input Buffer Circuit.....	20
2.8 Summary	21
Chapter 3 Design Methodology for the Layout Synthesis of I/O Pad Macros	23
3.1 Proposed Methodology	23
3.1.1 Overview of Shape Functions	26
3.1.2 Overview of Module Generators	28
3.2 Process Information.....	28

3.3	Floorplanning	30
3.3.1	Overview	30
3.3.2	Problem Definition	30
3.3.3	Pre-Placement	31
3.4	Routing	33
3.5	Compaction	34
3.6	Summary	34
Chapter 4	Shape Functions and Module Generators	35
4.1	MOS Transistors	35
4.1.1	Module Generator	35
4.1.2	Shape Function	36
4.1.3	Design Constraints	38
4.1.4	Typical Shape Functions and Example Layouts	39
4.2	Bipolar Transistors	46
4.2.1	Module Generator	46
4.2.2	Shape Function	48
4.2.3	Design Constraints	50
4.2.4	Typical Shape Function and Example Layouts	50
4.3	Diodes	54
4.3.1	Module Generator	54
4.3.2	Shape Function	55
4.3.3	Design Constraints	56
4.3.4	Typical Shape Function and Example Layouts	56
4.4	Rectangular Resistors	63
4.4.1	Module Generator	63
4.4.2	Shape Function	64
4.4.3	Design Constraints	65
4.4.4	Typical Shape Function and Example Layouts	66
4.5	Tiled Resistors	69
4.5.1	Module Generator	69
4.5.2	Shape Function	70
4.5.3	Design Constraints	71
4.5.4	Typical Shape Function and Example Layouts	71
4.6	Pad Opening	74
4.7	Lateral Silicon Controlled Rectifier (LSCR)	76
4.8	Snap-Back Transistor Generator	78
4.9	Summary	79
Chapter 5	Prototype Implementation	80
5.1	Design Concepts for the Synthesis Problem	80
5.2	Language of Construct	80
5.3	Abstraction	81
5.3.1	Data Abstraction	82

5.3.2	Example of Data Abstraction.....	83
5.3.3	Procedural Abstraction.....	85
5.3.4	Example of Procedural Abstraction.....	85
5.3.5	Variable Abstraction.....	88
5.4	Modular and Hierarchical Architecture.....	89
5.5	Module Generator Hierarchy.....	90
5.6	Module Generator Program Flow.....	91
5.6.1	Parameter Mapping and Parsing.....	91
5.6.2	Cell Geometry Computation.....	91
5.6.3	Translation and Rotation.....	93
5.6.4	Pin Alignment to Routing Grid.....	93
5.6.5	Generation of CIF or EDIF Code.....	93
5.7	Module Generators Run-Time and Memory Requirements.....	94
5.8	Shape Function Architecture.....	94
5.9	Shape Function Program Flow.....	94
5.10	Shape Function Run-Time and Memory Requirements.....	95
5.11	Experimental Coding Results.....	95
5.12	Summary.....	96
Chapter 6	Example Layouts.....	97
Chapter 7	Conclusions and Suggestions for Future Research.....	101
7.1	Summary.....	101
7.2	Contributions.....	102
7.3	Suggestions for Future Research.....	102
	Glossary of Terms and Symbols.....	103
Appendix A	Coding Convention.....	106
A.1	Naming of Variables.....	106
A.2	Module Generators and Shape Functions.....	107
A.3	General Conventions.....	108
Appendix B	Example Source Codes.....	109
Appendix C	Design Processes.....	114
	References.....	115

List of Figures

FIGURE 2.1	Typical layout template for narrow pad configuration.	4
FIGURE 2.2	Typical layout template for wide pad configuration.	5
FIGURE 2.3	Mirror MOS transistors in output drivers of CMOS' wide pad configuration.	6
FIGURE 2.4	Source of latch-up in CMOS bulk (figure extracted from [23]).	7
FIGURE 2.5	Example circuit for ESD protection.	9
FIGURE 2.6	Cross-sectional view of snap-back transistor.	10
FIGURE 2.7	LSCR layout and schematic representation.	11
FIGURE 2.8	Schematic of a typical CMOS-to-CMOS output buffer.	14
FIGURE 2.9	Schematic of a typical CMOS-to-CMOS input buffer.	15
FIGURE 2.10	Schematic of a typical bi-directional CMOS buffer.	15
FIGURE 2.11	Schematic of a typical CMOS-to-TTL output buffer.	16
FIGURE 2.12	Schematic of a typical TTL-to-CMOS input buffer.	17
FIGURE 2.13	Schematic of a typical TTL bi-directional buffer.	18
FIGURE 2.14	Schematic of a typical ECL output buffer.	19
FIGURE 2.15	Band-gap reference circuit for ECL output buffer.	19
FIGURE 2.16	Schematic circuit of a typical ECL input buffer.	20
FIGURE 2.17	Band-gap reference circuit for ECL input buffer.	21
FIGURE 3.1	Proposed layout synthesis flowchart.	24
FIGURE 3.2	Types of shape functions.	27
FIGURE 3.3	Pitch of interconnect objects in process with stacked and non-stacked interconnects.	29
FIGURE 3.4	Example layout solutions of relative pre-placement for CMOS output driver's transistors.	32
FIGURE 3.5	Example of inclusion of a guard band in a routing channel.	33
FIGURE 4.1	Abstract layout representation of fingered MOS transistors.	37
FIGURE 4.2	Shape function of PMOS transistors with $W_{fmin} = 5 \mu m$ and $W_{fmax} = 250 \mu m$	40
FIGURE 4.3	Shape function of NMOS transistors with $W_{fmin} = 5 \mu m$ and $W_{fmax} = 250 \mu m$	41
FIGURE 4.4	Synthesized layouts of PMOS transistors in a $0.8 \mu m$ BiCMOS process.	42
FIGURE 4.5	Synthesized layouts of PMOS transistors in a $1.2 \mu m$ CMOS process.	43
FIGURE 4.6	Synthesized layouts of NMOS transistors in a $0.8 \mu m$ BiCMOS process.	44
FIGURE 4.7	Synthesized layouts of NMOS transistors without a guard ring in a $1.2 \mu m$ CMOS process.	45
FIGURE 4.8	Abstract layout representation of fingered bipolar transistor.	48

FIGURE 4.9	Shape function of NPN bipolar transistors in a 0.8 μm BiCMOS process with a collector ring. $L_{E1mn}=L_{Emn}=L_{Eref}$, $L_{E1max}=L_{Ema}=40 \times L_{Eref}$	51
FIGURE 4.10	Synthesized layouts of bipolar transistor in 0.8 μm BiCMOS process.....	52
FIGURE 4.11	Synthesized NPN transistors in a 0.8 μm BiCMOS process without a collector contact ring.....	53
FIGURE 4.12	Abstract layout representation of a diode.	54
FIGURE 4.13	Shape function of p+/n diodes.....	57
FIGURE 4.14	Shape function of n+/p diodes.....	58
FIGURE 4.15	Synthesized layouts of p+/n diodes in a 0.8 μm BiCMOS process.....	59
FIGURE 4.16	Synthesized layouts of p+/n diodes in a 1.2 μm CMOS process.....	60
FIGURE 4.17	Synthesized layouts of n+/p diodes in a 0.8 μm BiCMOS process.....	61
FIGURE 4.18	Synthesized layouts of n+/p diodes in a 1.2 μm CMOS process.....	62
FIGURE 4.19	Layout and schematic representation of a rectangular resistor.	63
FIGURE 4.20	Shape functions of rectangular resistors.	67
FIGURE 4.21	Synthesized layouts for n-well resistors in a 0.8 μm BiCMOS process.....	68
FIGURE 4.22	Synthesized layouts for n-device well resistors in a 1.2 μm CMOS process.....	68
FIGURE 4.23	Abstract layout representation of tiled resistors connected in parallel with $n_r = 2$ and $n_c = 3$	69
FIGURE 4.24	Shape functions for a 200 Ω polysilicon tiled resistor connected in parallel.	72
FIGURE 4.25	Synthesized layouts of tiled resistors in a 0.8 μm BiCMOS process.....	73
FIGURE 4.26	Synthesized layouts of a four-connector pad opening.....	75
FIGURE 4.27	Synthesized layout of a LSCR.....	77
FIGURE 4.28	Synthesized layout for a snap-back transistor with $W_1 = 15 \mu\text{m}$	78
FIGURE 5.1	Fragmented Lisp codes describing examples of flavors.	84
FIGURE 5.2	Construction of diode diffusion polygon with procedural abstraction.	87
FIGURE 5.3	A typical software module.....	89
FIGURE 5.4	Module generators hierarchy.....	90
FIGURE 5.5	Module generator program flow.	92
FIGURE 6.1	Synthesized layouts of bi-directional CMOS I/O buffers constructed in a 0.8 μm BiCMOS process.	99

FIGURE 6.2	Synthesized layouts of bi-directional I/O buffers constructed in a 1.2 μm CMOS process.	100
-------------------	---	------------

Chapter 1

Introduction

1.1 Motivations

The layout synthesis of I/O pad circuits has received little attention by the design automation community because of the challenging design space resulting from merging complex physical shapes with intricate electrical behavior. Numerous layout considerations must be made to provide adequate protection against electrostatic discharge (ESD) as well as immunity against latch-up. These requirements are usually coupled with process parameters and physical layout constraints. As a result, current layout design of I/O pad macros is still typically hand-crafted.

Since I/O circuits exist in all integrated circuits (ICs), automating the layout of these circuits has immense potential for a reduction of development time and cost. Key motivations that this research is emerging from include an ability to construct I/O pads for various layout configurations, a framework to design buffer circuits for interfacing signals between different logic families, and a layout tool to parametrize I/O pad circuits.

1.2 Objectives

The first objective of this thesis is to provide a description of I/O pad macro circuitry which is adequate for understanding the problem involved in the design of I/O pads: electrical and physical constraints. The second objective is to outline a design methodology for a layout synthesis system for I/O pad macros. The third objective and perhaps the most important goal of this research is to describe two key components in the proposed methodology: module generators and shape functions. Some of the key elements that this thesis investigates are the layout issues associated with latch-up, electrostatic discharge (ESD), floorplanning and routing as well as process information in the context of designing module generators for I/O pads. The last objective is to show experimental

results to demonstrate the functionality and applicability of the module generators and shape functions.

1.3 Thesis Outline

Chapter 2 reviews the characteristics of CMOS, TTL and ECL logic families along with a brief description of selected buffer circuits for CMOS, TTL and ECL compatible signals. A detailed description of typical pad configurations as well as a background on ESD and latch-up are also presented in this chapter. Chapter 3 describes the proposed design methodology for the layout synthesis of I/O pad macros. The issues associated with process information, floorplanning, routing and compaction are considered for the design methodology are discussed in this chapter. A detailed description of the shape functions and corresponding generators follows in chapter 4. Example layouts synthesized by the module generators are also presented. Chapter 5 discusses the prototype implementation of the shape functions and related generators. Example layout of pad macros being produced with the help of the module generators are presented in chapter 6. Finally, chapter 7 summarizes the thesis and proposes some future work.

Chapter 2

Design of Standard I/O Pad Macros

This chapter describes the design and layout issues associated with the design of CMOS, TTL and ECL compatible pad macros. The relevant issues include (i) physical constraints imposed by different layout template, (ii) immunity against latch-up, (iii) electrostatic discharge (ESD) protection, and (iv) compatibility between different logic families. The first section of the chapter presents the typical layout configurations for the pad macros. The next section gives a background on latch-up and ESD and proposes some layout guidelines for prevention and protection purposes. Finally, the last section gives a brief description of typical buffer circuits selected for CMOS, TTL and ECL compatible pad macros.

2.1 Typical Pad Configurations

There are two typical layout configurations for I/O pad macros: narrow and wide. The narrow configuration is used in circuits with very large pin count (pad limited) and the wide configuration is found in circuits with relatively few I/O pins (core limited). For both cases, the I/O pad macros serve the double function of interfacing with external signals and distributing supply voltages around the periphery of the chip.

2.1.1 Narrow Pad Configuration

The layout template of the narrow pad is a tall and rectangular cell with the pad opening located near the top. The output drivers and protection circuitry are typically near the center of the cell whereas the control logic block and pre-drivers are near the bottom. Separate power rails are usually used to supply the pad macro and the chip's core circuitry. This is due to the large demand for supply current generated by the large number of pads as well as to decouple cross talk noise. Figure 2.1 illustrates a typical layout template for a narrow pad configuration.

FIGURE 2.1 Typical layout template for narrow pad configuration.

A pair of V_{DD} and V_{SS} *core* supply rails (for the chip's logic core) as well as a pair of V_{DD} and V_{SS} *ring* supply rails (exclusively for I/O pads) are normally superimposed on top of the I/O pad circuit elements. In addition, a V_{SS} ring is usually found near the top edge of the cell. All PMOS and NMOS transistors, except those in the output drivers, are inside the cell. All PMOS and NMOS transistors, except those in the output drivers, are inside the n-well and p-well at the lower end of the pad respectively. These wells can be merged between adjacent cells in order to form concentric rings around the periphery of the chip. Guard bands which are used to separate the n and p wells for latch-up protection can also be strung up in a similar fashion.

Because the constraints on the position and dimensions of the power rails, the height of the pad is usually restricted by these busses. The width of the cell, on the other hand, can vary but its minimum size is usually bounded by the minimum pitch distance between the pad openings.

For the case of ECL compatible pads, the layout template is slightly different from that of CMOS and TTL as no NMOS or PMOS transistors are used. As such, there is no regular structure of n-well and p-well because only NPN transistors are used. The general position of the power rails, however, remains the same to allow ECL compatible pads to be placed adjacent to any of the CMOS or TTL I/O cells.

2.1.2 Wide Pad Configuration

In the wide pad configuration, there is usually a single pair of supply rails which is shared by all circuits on the chip. The width of these power rails is typically larger than those found in a narrow configuration. Figure 2.2 shows a typical layout template for a wide pad configuration.

FIGURE 2.2 Typical layout template for wide pad configuration.

Wide pad macros are usually square with the pad opening typically placed near the middle of the layout. The PMOS and NMOS output drivers in a CMOS compatible pad are frequently situated on opposite sides of the pad opening for immunity against latch-up. For this reason, the MOS devices type in the output driver of alternate pads are mirrored as illustrated in Figure 2.3. In addition, as in the case of the narrow configuration, a guard band is placed between the n and p wells to further reduce the susceptibility to latch-up. Finally, the protection circuitry is typically located near the top while the control logic is usually found near the bottom of the cell.

Similar to the narrow pad, the height of the wide pad depends on the position and dimensions of the power busses. The minimum width is constrained by the pitch distance consisting of the sum of the pad opening and the MOS transistors in the output driver.

FIGURE 2.3 Mirror MOS transistors in output drivers of CMOS' wide pad configuration.

2.2 Latch-Up Prevention

2.2.1 Overview of Latch-Up

Latch-up in CMOS (or CMOS portion of BiCMOS) circuits is due to the presence of parasitic bipolar devices forming a *pnpn* structure, also called Semiconductor Controlled Rectifier (SCR). Figure 2.4(a) shows a cross-sectional view of the parasitic npn and pnp bipolar transistors with the equivalent lumped circuit illustrated in Figure 2.4(b) [23]. Normally, these parasitic transistors are biased off. However, positive feedback can occur if the well current, I_w , or the substrate current, I_s , is sufficiently large to forward bias the base-emitter junction of either the npn or pnp transistors since the collector of one transistor feeds the base of the other. Such conditions can be triggered by spurious noise, voltage overshoots or ESD resulting from chip manipulation. When the parasitic transistors are biased active, they form a loop with a gain equal to the product of $\beta_{nnp}\beta_{pnp}$, where β_{nnp} and β_{pnp} are the current gains for the npn and pnp transistors, respectively. If

this product gain exceeds unity, it causes a sustained high current flowing between the V_{DD} and V_{SS} power rails. This phenomenon is known as latch-up and can cause permanent damage to the circuit.

(a) Cross-sectional view of parasitic bipolar transistors and resistors in an n-well CMOS structure.

(b) Equivalent lumped circuit.

FIGURE 2.4 Source of latch-up in CMOS bulk (figure extracted from [23]).

2.2.2 Layout Guidelines for Prevention Against Latch-Up

Several measures have been suggested to prevent latch-up [1, 7, 23] and can be classified into two categories: latch-up prevention through processing and prevention through layout techniques. Their objectives are to (1) reduce the well and substrate resistances (R_w and R_s), (2) limit the lateral minority carrier current, and/or (3) reduce the current gain of the bipolar transistors. Some of processing techniques used for latch-up prevention include retrograded well, gold doping, and trench isolation [7]. These measures can not always be realized because of the cost involved. Prevention measures against latch-up that a designer can control are best achieved with layout techniques. Typical layout considerations used for I/O pad macros include [15]:

- i) allowing sufficient space between active regions, especially between the PMOS and NMOS transistors in the output drivers of CMOS compatible pads;
- ii) adding p+ or n+ guard bands (tied to V_{ss} and V_{DD} respectively) around or between large current conducting devices such as output drivers and diodes to collect excess minority carriers being injected to the well or the substrate;
- iii) connecting by abutment all source diffusions and wells which must be tied to the supply rails with split contacts.

2.3 Electrostatic Discharge (ESD) Protection

2.3.1 ESD Protection Circuit

Electrostatic discharge (ESD) is a significant cause of device failure especially in circuits with large transient current such as I/O pad circuits. It results in gate oxide puncture, fusion of diffusion or polysilicon lines, etc., and can damage or destroy the IC. Figure 2.5 shows an example ESD protection circuit that includes a p+/n diode, a snap-back transistor, a resistor and a lateral silicon controlled rectifier (LSCR) [11].

FIGURE 2.5 Example circuit for ESD protection.

The role of the p+/n diode is to limit the voltage applied to the core circuitry (V_{in}) when the pad voltage exceeds $V_{DD} + V_{diode}$ where V_{diode} is the forward voltage across the diode. Figure 2.6 illustrates the cross-sectional view of the snap-back transistor with parasitic elements indicated by the dashed lines. A bipolar NPN transistor is formed by the source, substrate and drain. The function of the snap-back transistor as an ESD protection device is two-fold. When a high current flowing into the drain reaches certain threshold, the parasitic NPN transistor goes into a mode called snap-back characterized by a negative resistive. A current path exists between the snap-back transistor's drain and source, draining the transient current to protect the core circuitry. Such condition takes place when the power supply of the circuit (V_{DD}) increases and pad voltage also rises above the normal operating voltage of V_{DD} . The second function of the snap-back transistor is to provide protection when V_{in} falls below V_{SS} . The base-collector junction diode of the

NPN transistor becomes forward biased if the drain potential is sufficiently lower than the substrate.

FIGURE 2.6 Cross-sectional view of snap-back transistor.

Resistor R limits the peak current that flows into the diode or the snap-back transistor when either of them is conducting. In addition, the parasitic diode between the n-well resistor and the p-substrate serves as protection device from abnormal transient voltages below the negative rail (V_{SS}). The resistor, in conjunction with the input capacitance, C , leads to an RC time constant which must be watched in high speed circuits.

The function of the LSCR is to collect parasitic charges from the pad opening to prevent a discharge to the core circuitry. Figure 2.7 shows a cross-sectional view of the LSCR and its equivalent schematic representation. It is designed based on the topology and concept of a Silicon Controlled Rectifier (SCR) circuit [22]. The device is on when a sufficient amount of charge is accumulated at the pad opening, inducing a current across the n-well/p-substrate parasitic capacitor C_{np} . If this current is large enough to turn either Q_1 or Q_2 on, a conduction path occurs between the pad opening and V_{SS} . As a result, the excess charge is drained from the pad opening. The voltage at which the LSCR is triggered depends on C_{np} , R_w or R_s . These parameters can be controlled by the sizes and separation features of the diffusion polygons (pnpn structure).

(a) Cross-sectional view of a LSCR.

(b) Equivalent schematic circuit of a LSCR.

FIGURE 2.7 LSCR layout and schematic representation.

2.3.2 Layout Guidelines for ESD Protection

In addition to the inclusion of ESD protection devices in the pad macro circuit, a number of layout guidelines are usually applied to increase the likelihood that a circuit will survive an ESD hit. These guidelines are usually provided by process vendors. The guidelines used for the prototype layouts in this research include [15]:

- i) using greater than minimum separation between selected layout features in order to better distribute large currents (e.g. spacing between gate and source/drain diffusion contacts);
- ii) restricting the minimum size of any transistor connected directly to an I/O pad to be above a certain threshold to avoid burnout problem;
- iii) bevelling the corners of rectangular layout features carrying large currents in order to limit current crowding.

2.4 Mixed Levels Signal Compatibility

The presence of mixed signals in today's applications is becoming more and more frequent. In fact, many high speed applications have high-density, low-power CMOS devices in the core and high-speed, high-drive capability bipolar components in the periphery to interface with TTL or ECL signals [25].

The key issues associated with the design of TTL to CMOS interface circuits are a mismatch in logic levels and switching threshold. In worst cases, TTL devices have an output swing between 0.4 V and 2.4 V. The TTL minimum output high logic voltage (2.4 V) is normally not high enough for typical CMOS gate to fully recognize a high input logic. This is because the switching threshold of CMOS devices is 2.5 V for a power supply of 5 V whereas that of TTL logic signals is typically 1.4 V. The main issue associated with CMOS to TTL signal conversion is current sinking. CMOS devices must be able to sink 16 mA (assuming the TTL gate is designed to source 1.6 mA per fan-out) from a TTL gate when the TTL's input voltage is held low (i.e. approximately 0.4 V).

Due to a small voltage swing and negative switching voltages, the design of ECL interface circuits usually involves level shifting and amplification. Typical ECL devices have an output swing approximately between -1.7 V and -0.9 V and a switching threshold in the vicinity of -1.3 V at room temperature. Another factor that needs to be considered in the design of ECL buffers is compensation against the effect of temperature variations on the circuits. This can be achieved with a temperature compensated voltage reference network in the form of a band-gap reference voltage [17] or an automated bias control (ABC) circuit [10].

2.5 CMOS Compatible Buffer Circuits

2.5.1 CMOS-to-CMOS Output Buffer Circuit

Figure 2.8 illustrates the schematic of a typical CMOS-to-CMOS output buffer circuit [11]. The output drivers are typically a pair of very large complementary MOS devices used for the pull-up and pull-down. Complementary devices are used to provide full logic swing but are more susceptible to latch-up. It is not unusual for these transistors to have a width/length aspect ratio of 500/1 or even 1000/1 to provide a desired drive capability. Apart from the considerations made for ESD protection as described in Section 2.3, the output driver is usually laid out into a number of transistors connected in parallel. This layout strategy provides a better distribution of current density. The pre-drivers play a key role in limiting the slew rate of the output signal, and hence the ground bounce effect. The logic block in this simple example controls the tri-state operation of the output pad but could include much more complex functions such as boundary scan circuitry and/or storage elements.

FIGURE 2.8 Schematic of a typical CMOS-to-CMOS output buffer.

2.5.2 CMOS-to-CMOS Input Buffer Circuit

Figure 2.9 shows the schematic of a typical CMOS-to-CMOS input buffer circuit [11]. The input buffer is comprised of two inverters connected in series, with increasing size, to buffer the input signals. The other circuit components are used for electrostatic discharge (ESD) protection.

FIGURE 2.9 Schematic of a typical CMOS-to-CMOS input buffer.

2.5.3 CMOS-to-CMOS Bi-Directional Buffer Circuit

By merging an input and output buffer, a bi-directional buffer circuit can be constructed as shown in Figure 2.10 [11]. The layout considerations made for latch-up and ESD protection, and layout configurations are similar as those found in an input or output buffer.

FIGURE 2.10 Schematic of a typical bi-directional CMOS buffer.

2.6 TTL Compatible Buffer Circuits

2.6.1 CMOS-to-TTL Output Buffer Circuit

Figure 2.11 shows a typical CMOS-to-TTL output buffer [11]. It is similar to its CMOS-to-CMOS counterpart with a key difference being the PMOS pull-up device of the output driver is replaced by an NMOS transistor. An Enable signal is also available for tri-state operation; the logic gates are changed accordingly to maintain the functionality. The NMOS/NMOS configuration is selected because TTL devices do not require the input voltages to swing from rail to rail (i.e. between ground and 5 V). In addition, this configuration provides a better immunity against latch-up resulting from the absence of an n-well. The pull-down NMOS transistor is sized to satisfy the 16 mA current sinking when the output is low. In order to have symmetrical rising and falling output transients, the NMOS devices in the output driver are approximately the same size [11]. Similar to the CMOS compatible buffers, these NMOS devices are constructed by connecting a number of smaller transistors in parallel.

FIGURE 2.11 Schematic of a typical CMOS-to-TTL output buffer.

2.6.2 TTL-to-CMOS Input Buffer Circuit

Figure 2.12 illustrates a typical TTL-to-CMOS input buffer circuit [11]. It is designed to minimize the effect of stand-by power dissipation. The key feature of this circuit is a pair of inverters connected together with the first inverter having a reduced V_{DD} and the second inverter having its input and output connected through a feedback device. Transistors Q_1 , Q_2 and Q_3 provide a voltage drop equivalent to approximately three diode voltages below V_{DD} . Assuming V_{DD} is 5 V and a diode voltage drop is 0.7 V, the reduced V_{DD} at point A is about 2.9 V. This reduced power supply results in smaller input voltage levels at the first inverter with $V_{IL}(\text{max}) = 0.87$ V (i.e. 30% of 2.9 V) and $V_{IH}(\text{min}) = 2.03$ V (i.e. 70% of 2.9 V). As a result, the stand-by power dissipation problem of the first inverter is eliminated.

The role of the second inverter is to amplify the output of the first inverter to provide a full CMOS voltage swing. Since the maximum output voltage of the first inverter is approximately 2.9 V, the second inverter is exposed to stand-by power dissipation. The feedback transistor M_3 is added to remedy this problem. When the input falls, point B is pulled up and the output starts to fall until it is low enough to turn M_3 on, pulling point B towards V_{DD} to turn M_4 off.

FIGURE 2.12 Schematic of a typical TTL-to-CMOS input buffer.

2.6.3 TTL Bi-Directional Buffer Circuit

The TTL bi-directional buffer circuit combines a TTL-to-CMOS input and a CMOS-to-TTL output buffer and is shown in Figure 2.13 [11].

FIGURE 2.13 Schematic of a typical TTL bi-directional buffer.

2.7 ECL Compatible Buffer Circuits

2.7.1 ECL Output Buffer Circuit

Figure 2.14 shows the schematic of an ECL-to-ECL output buffer [11, 17]. It features a differential amplifier being driven by a differential signal. The output stage consists of an emitter-follower driving an external standard load of $50\ \Omega$ with one end connected to -2 V . The circuit is powered with its V_{EE} and V_{CC} connected to ground and 5 V respectively. By this way, the ECL buffer can fit into similar layout template of its CMOS and TTL counterparts. Figure 2.15 depicts a band-gap reference circuit [11, 17] designed to compensate the effect of temperature variations on the circuit when the input is high. The cross-coupled network consisting of D_1 , D_2 and R_3 , on the other hand, provides necessary temperature compensation when the input is low [11, 17].

FIGURE 2.14 Schematic of a typical ECL output buffer.

FIGURE 2.15 Band-gap reference circuit for ECL output buffer.

2.7.2 ECL Input Buffer Circuit

The topology of the ECL-to-ECL input buffer circuit, shown in Figure 2.16 [11, 17], is similar to that described for the output buffer. Since it accepts a single ended signal as input, a reference voltage is applied to the base of Q_2 . As a result, the band-gap reference circuit is modified in order to supply two constant voltages: V_{ref} and V_{CS} , where V_{ref} is the reference voltage and V_{CS} is the controlling voltage for the differential amplifier current source. The corresponding band-gap reference network is illustrated in Figure 2.17 [11, 17].

FIGURE 2.16 Schematic circuit of a typical ECL input buffer.

FIGURE 2.17 Band-gap reference circuit for ECL input buffer.

2.8 Summary

This chapter described a number of issues associated with the design of standard I/O pad macros. The key problems are multiple pad configurations, prevention against latch-up, ESD protection and mixed level signals. Due to these issues, the layout synthesis of I/O pad macros requires a number design considerations which can be summarized as follows.

The circuit elements in a I/O buffer circuit are a mixture of standard logic circuits and special devices. Standard logic circuits include the tri-state control logic, pre-drivers and input buffers. These elements can be laid out using normal layout techniques and can be synthesized with a leaf cell generator. Special devices consist of fingered transistors in an output driver (MOSFET and bipolar), diodes, LSCR, resistor (rectangular and tiled configuration), snap-back transistor and pad opening. In contrast to standard logic circuits, the layout of these devices requires special considerations to support all requirements for

ESD protection, immunity against latch-up as well as other physical constraints. In addition, there is a need to consider more than one pad opening in buffer circuits such as the ECL buffers. Furthermore, the synthesized cells must be flexible to accommodate different pad configurations, namely the narrow and wide configurations. As a result, the synthesis system for these special devices needs to provide a great deal of control from the user. For example, a greater than minimum spacing may be required for ESD protection, repeated features may be necessary for a given macro cell, etc. The proposed strategy for these requirements uses module generators and shape functions. The details of these components and their implementation are described in Chapters 4 and 5, respectively.

The design of a floorplan system for the pad macros also needs special considerations. The floorplan algorithm needs to superimpose supply rails on top of macro circuits. In addition, certain pre-placement features may be critical to support protection against ESD and latch-up. For example, the floorplan may require to include additional guard bands and/or guard rings for special latch-up requirements. The next chapter deals with the issues involved in the design of the floorplan as well as other relevant components such as process information, routing and compaction.

Chapter 3

Design Methodology for the Layout Synthesis of I/O Pad Macros

This chapter presents a design methodology for the layout synthesis of I/O pad macros. As discussed in Chapter 2, the design space for layout synthesis of I/O pad macros is restricted by a number of design constraints. These requirements are concerned with the ability to control the sensitive layout features necessary for ESD protection and immunity against latch-up as well as to generate specific or repeated features. The proposed solution for these requirements is module generators. In addition, the physical constraints imposed on the individual circuit elements require a mean of selecting the aspect ratio for each building block. As such, the total area of a pad macro can be minimized by selecting proper aspect ratios for each layout module. These requirements can be satisfied with the help of the shape functions.

Other relevant factors that need to be addressed are the specific issues related to process information, floorplanning, routing and compaction. There are two objectives associated with the discussion of these issues. The first objective is to identify the attributes that need to be incorporated in the module generators and shape functions. The second objective is to provide a general specification (set of requirements) for the floorplan algorithm devised for the layout synthesis of I/O pad macros, without actually providing a solution.

3.1 Proposed Methodology

The design methodology for the layout synthesis of I/O pad macros is illustrated in Figure 3.1 and can be summarized as follows. First, a collection of special purpose module generators is used to define the detailed layout of individual building blocks such as transistors, resistors, diodes, etc. For each building block, a shape function (area model) is

FIGURE 3.1 Proposed layout synthesis flowchart.

developed to summarize the range of feasible designs both in terms of layout area and aspect ratio assuming rectangular blocks. Each point on a shape function corresponds to a possible layout that could be produced by a module generator. Each area model is intimately coupled with its corresponding generator in order for the model to accurately reflect the feasible solution space of the generator.

Based on the area models as well as constraints on the overall aspect ratio of the complete pad macro, a floorplan is designed for the circuit that selects the aspect ratio and imposes other physical constraints on the individual building blocks. This process is similar to the VLSI floorplanning process with flexible blocks, with the key distinction being that the nature of the shape functions can vary widely between circuit elements. The layout features (i.e., power rails, wells, etc.) constitute the layout template onto which the building blocks must subsequently be placed and routed. The outcome of the floorplan process is a placement solution in which the relative position and the overall layout dimensions of each layout module are known. The position of the modules is determined in a manner such that enough room is left between them for routing.

Following the floorplan, the module generators construct the detailed layout geometries for the layout modules and place them on a symbolic routing grid. The generators also synthesize I/O pins on the modules to align them with the routing grid. Given the relative position of the pins and their connectivity within the pad circuitry, a detailed routing is performed. Next, a compaction process is carried out to eliminate any white space introduced by floorplanning or routing (between wires or between wires and modules). The last step consists of mapping the symbolic wires into physical coordinates.

3.1.1 Overview of Shape Functions

A shape function is composed of a pair of equations which summarize the physical dimensions of feasible layouts of a circuit element. Each point on a shape function represents a solution that constitutes the bounding contour, assuming rectangular blocks, that is large enough to accommodate the layout device. The shape function equations, width (X) and height (Y), can be derived from a set of electrical specifications and latch-up and ESD protection requirements as well as physical constraints imposed on the layout. More specifically, these equations are constructed by modelling the required area from accessible parameters such as transistor count, size of active area, width of guard rings, and so on. The generalized form of a shape function can be described with Equations (3.1) and (3.2) where $parm1$, $parm2$, ..., $parmN$ are some of the accessible parameters.

$$X = f(parm1, parm2, ..., parmN) \quad (3.1)$$

$$Y = g(parm1, parm2, ..., parmN) \quad (3.2)$$

There are two types of shape function: discrete and continuous [13]. The solution of a discrete shape function denotes a finite number of layout alternatives whereas that of its continuous counterpart represents an infinite number of layout possibilities. Figure 3.2 illustrates examples of a discrete and a continuous shape function. SpecsA and SpecsB in these simple examples represent two possible sets of specifications imposed on a given layout object.

Usually, the design space of a layout object is delimited by a number of constraints. These constraints can be user specified or simply limitations of the target process. For example, the number of layout solutions can be reduced to only those with boundaries within a set of allowable dimensions, namely between X_{min} and X_{max} , and Y_{min} and Y_{max} respectively. Another example of design constraint are the limits imposed on certain electrical properties of a circuit element. The limiting values for a set of specifications can be represented by SpecsMin and SpecsMax as illustrated in Figure 3.2. The layouts A and B shown in Figure 3.2(b) denote two extreme solution instances of a device characterized by SpecsA. Layout C is an example realization selected by the floorplan.

FIGURE 3.2 Types of shape functions.

3.1.2 Overview of Module Generators

Module generators are programs which create the layout geometries for a discrete device or a functional building block based on a set of user and process parameters. Types of parameters are those affecting the layout functionality, performance, size, shape, electrical characteristics, pin assignment as well as protection against ESD and latch-up.

The design of module generators must address the issue of process independence. In addition, the generators must be flexible to control the input parameters in order to provide layouts with desired features. Furthermore, considerations for issues associated with process information, floorplanning and routing to be described in the subsequent sections must be incorporated in the module generators. By this way, the generators can be integrated into existing floorplanning and routing systems with minor modifications to the algorithms.

3.2 Process Information

Process information can be separated into two major components: the front end and the back end. The front end information include polygons with layers defining the devices, wells, and guard rings or guard bands (everything below polysilicon material). The back end information consist of interconnect layers, namely contact, metal1, via, metal2, etc. In general, the back end does not vary very much. The key information that changes is the number of metal layers and the pitch of each one. The only exception to this is stacked vias.

In non-stacked interconnect processes, spacings among contacts, vias, and topvias must be large enough to ensure the placement of contacting elements of lower layers (e.g. contacts) does not violate the clearance rule from higher interconnect layer elements (e.g. vias). This implies that the module generators must be designed such that the determination of the pitch for contacts, vias or topvias is a function of the layers to be interconnected. For example, if a diffusion region is to be connected to a metal2 wire, three layers are required. These layers are contact, metal1 and via. In this case, the module generator must include the size of a via when it determines the pitch for the contacts

placement, and vice versa. An illustration of this problem is shown in Figure 3.3. Clearly, the pitch of interconnect objects in processes that allow stacked interconnects is smaller than those which do not.

(a) Process with stacked interconnects.

(b) Process with non-stacked interconnects.

FIGURE 3.3 Pitch of interconnect objects in process with stacked and non-stacked interconnects.

3.3 Floorplanning

3.3.1 Overview

Floorplanning is concerned with the placement of rigid and flexible building blocks such that the total area used by both the building blocks and the interconnect is minimized. Typically, the small MOS devices associated with the control logic, pre-drivers, and input buffers may be grouped to form locally optimal chains of MOS transistors, and each of them is treated as a module in the placement area. These modules can be constructed with the help of a leaf cell synthesis system [12]. Circuit elements whose layout is more critical to ESD and latch-up such as the output drivers, diodes, etc. are handled by special purpose module generators [15].

In contrast to the classical floorplanning problem, the floorplan algorithm devised for I/O pad macros must allow the power rails to overlap any layout blocks, and must have the ability to satisfy certain pre-placement requirements. The power rails can overlap other blocks because they are usually defined with the highest metal of the target process. The internal wiring of the module generators, on the other hand, is restricted to lower metal interconnect layers. As such, the wiring between a power rail and the pad's building blocks can be reduced, hence providing a smaller layout area as well as signals delay.

3.3.2 Problem Definition

The problem of floorplanning for I/O pad macros can be stated as follows.

Given:

- i) n modules specified by their shape function,
- ii) a netlist describing the interconnections between these modules,
- iii) a set of pre-placement constraints, and
- iv) the target height of the pad,

determine:

- i) the dimension of each module, and
- ii) the relative position of each module,

such that:

- i) there is no overlap between the modules with the exception of the power rails,
- ii) all the physical constraints for ESD and latch-up protection are satisfied, and
- iii) the resulting layout has minimum width.

3.3.3 Pre-Placement

The pre-placement problem in a pad macro can be categorized into two types: absolute and relative positioning. Absolute positioning is necessary for supply rails and device wells. As described in Section 2.1, the number and position of the power rails are usually fixed and depend on the pad configuration. The device wells of adjacent pads are strung up together to form concentric well ring around the periphery of the chip. The absolute position of the power rails and device wells constitute the layout template onto which the remaining building blocks are placed and routed.

Relative pre-placement deals with the positioning of building blocks in a specified region. The only requirement for the placement of these building blocks is that they must be in the specified region but are not constrained to an absolute vertical or horizontal position. For example, in the case of the wide configuration, the pull-up PMOS and pull-down NMOS transistors of a CMOS output buffer circuit are usually situated in the region on each side of the pad opening for a better immunity against latch-up. This is illustrated in Figure 3.4 where the placement of these transistors is restricted to be in the dashed regions.

FIGURE 3.4 Example layout solutions of relative pre-placement for CMOS output driver's transistors.

One important factor which must be considered during floorplanning is the inclusion of guard bands and/or guard rings to enhance immunity against latch-up. In some layout blocks such as the MOS transistors in the output drivers, guard bands and/or guard rings are already integrated in the layout module. Other blocks such as the logic block or the bipolar transistors may or may not have a pre-laid guard bands and/or guard rings in the synthesized layout. If two layout blocks of opposite well types are placed nearby each other, a guard band or a guard ring should be added if no guard is included around one of them. The placement of these guard bands or guard rings constitute a relative pre-placement case. Figure 3.5 depicts an example where a guard band is inserted in the routing channel (denoted by the dashed region) between the n-well surrounding the pull-up PMOS transistor of an output driver and the p-well of the logic block.

Every placement permutation that a floorplan algorithm explores must satisfy the pre-placement constraints. If a given solution fails a pre-placement condition, subsequent solutions that could be derived from the latter one are discarded. As a result, the placement space that a floorplan algorithm must explore can be reduced.

FIGURE 3.5 Example of inclusion of a guard band in a routing channel.

3.4 Routing

The objective of routing is to connect pins with a same net such that a given measure of wiring cost or routability is minimized. This is usually done in two steps, referred to as global routing and detailed routing. In both steps, routing is done on a symbolic routing grid instead of an absolute coordinate grid in order for the router to decouple any knowledge about a specific layout process from its algorithm. In order for the router to interconnect the circuit elements, it is necessary to align the pins of the layout blocks to the symbolic routing grid. If a pin of a layout block is not on a routing grid, the associated module generator adds a wire in order to snap the pin terminal to the nearest grid. The width of the additional wire must be larger than or equal to that of the pin terminal. In this way, the current density flowing in or out from the pin is not too large for the additional wire. A pin may be required to align to an horizontal or to a vertical grid, or both. It depends on the orientation of the interconnect wire. If the wire runs horizontally, then the termination pin must snap to an horizontal grid. Otherwise, it must align to a vertical grid. Since the module generators do not have the knowledge how the interconnect wires are oriented, it is desirable to map the pins to both horizontal and vertical grids.

3.5 Compaction

Upon completing routing, a compaction process is required to remove any white space introduced by the floorplan and/or the router. Since the building blocks synthesized by the module generators are rigid in size at this stage, compaction is not hierarchical. The internal layout geometries of the modules are left untouched, only that they are pushed closer to each other wherever possible. Since ESD and latch-up are important factors in the layout of I/O pad macros, special control on the compaction algorithm is required to allow the implementation of selected layout guidelines as described in Sections 2.2.2 and 2.3.2.

3.6 Summary

This chapter described a number of issues associated with the design methodology for the layout synthesis of I/O pad macros. A contribution of this chapter is a description of the key requirements for a floorplan algorithm. These requirements are concerned with (i) the ability to overlap the power rails on individual circuit elements, and (ii) pre-placement specifications which include absolute and relative positioning.

Another relevant contribution is the identification of key requirements for the module generators and shape functions. The implementation strategy for the module generators must address (i) the ability to control specific layout features, (ii) the flexibility to accommodate different layout processes, and (iii) the capability to interface with other components such as floorplan, router and compactor. Similar requirements for the shape functions are also necessary but with one exception. The shape functions are only concerned with the sizes associated with the variants in the layout features and layout processes. The next chapter is devoted for the description of the module generators and shape functions. The detailed implementation strategy is described in Chapter 5.

Chapter 4

Shape Functions and Module Generators

This chapter describes in detail : shape functions and module generators for the individual circuit elements. These include the output driver transistors, diodes, resistors, snap-back transistor, LSCR and pad opening. Sample layouts are included based on design rules for a 1.2 μm CMOS and a 0.8 μm BiCMOS process in order to demonstrate the portability of the generators.

4.1 MOS Transistors

4.1.1 Module Generator

The large MOS transistors in the output drivers are constructed by connecting a number of smaller transistors in parallel. These transistors are referred to as fingers and can be laid out by themselves or in pairs with the drain of one transistor connected to the source of the next. For a given overall aspect ratio W/L , n fingers each with $W_f = W/n$ are required. Clearly, there are many combinations of W_f and n that will yield the same W/L with a different aspect ratio W_f/L . Because n is an integer, the shape function of fingered MOS transistors is discrete. The parameters that the MOS transistor generator accepts to synthesize corresponding layout geometries are summarized below with the name of the input parameters being emphasized within parenthesis:

1. Transistor type (*MOSType*): PMOS or NMOS;
2. Transistor width and length (W and L): the length L is constant for a given layout process;
3. Number of fingers (n): the width of each fingered transistor is W/n ;
4. Finger configuration (*FingerConfig*): pair or single. If single transistors exist, the position for each single transistor is also given in this parameter;
5. Split contacts are used or not (*SplitCont*): split contacts are used to tie the substrate/well and the transistor's drain/source to a known potential to reduce susceptibility to latch-up;

6. Guard ring is required or not (*GuardConfig*): a guard ring may be required to increase the immunity against latch-up;
7. Width of guard ring (*GuardWidth*): this parameter depends on latch-up;
8. Number of contact columns (*ContCols*): this number determines the size of the diffusion area. More than minimum number of contacts may be required for ESD protection;
9. Size of contacts used (*ContSize*): this parameter depends on ESD and current density;
10. Horizontal and vertical contact spacings (*HVContSpa*): the larger the contact spacing is, the less susceptible the layout is subjected to ESD hit due to a larger resistance between the contacts.
11. Gate-contact spacing (*GateContSpa*): separation between the contacts and the polysilicon gate. This parameter depends on ESD protection.
12. Edge diffusion contact overlap (*EdgeContOvl*): the overlap value of an edge of a diffusion polygon on the contacts. This parameter depends on ESD protection requirement.
13. Position of gate connector (*GateConPos*): at the top edge or bottom edge. The position of the gate connector is configured to be closer to the terminals to which it is to be connected.
14. Number of contacts in the gate connectors (*GateContNum*): this parameter is determined by the width of the wire connecting to the gate connectors;
15. Netlist with pin positions (*Netlist*);
16. Layer of pins (*PinLayer*): if the layer of the interconnect wire is known, the layer of the pins is matched to that of the wire;
17. Position (*Position*) and orientation (*Orientation*) of the final layout: variable *Position* represents the center coordinates of the final cell within the floorplan.

4.1.2 Shape Function

Figure 4.1 illustrates an abstract layout representation of a fingered MOS transistor. The width and height, X and Y , of the fingered MOS transistors are derived as follows:

$$X = X_a + 2X_{gr} \quad (4.1)$$

$$Y = Y_a + Y_{gr1} + Y_{gr2} \quad (4.2)$$

where X_a and Y_a are the horizontal and vertical dimensions of the active area. Similarly, X_{gr} , Y_{gr1} and Y_{gr2} are the distances between the edges of the active area and the outer edges of the guard ring. The only distinction between Y_{gr1} and Y_{gr2} is the position of the gate connector and the number of contacts in that connector. This connector ties the gates to the pre-driver circuit.

FIGURE 4.1 Abstract layout representation of fingered MOS transistors.

The parameters in Equations (4.1) and (4.2) can be expanded as follows:

$$X_a = f(L, \text{MOS}type, n, \text{FingerConfig}, \text{ContCols}, \text{ContSpa}, \text{GateContSpa}, \text{ContSize}, \text{SplitCont}, \text{EdgeContOvl}) \\ \cong nK_1 + K_2 \quad (4.3)$$

$$Y_a = f(W, n) = W_f + K_3 = W/n + K_3 \quad (4.4)$$

$$X_{gr} = f(\text{MOS}type, \text{GuardConfig}, \text{GuardWidth}, \text{ContSpa}, \text{ContSize}) + K_4 = K_4 + K_5 \quad (4.5)$$

$$Y_{gr1} = f(\text{MOS}type, \text{GuardConfig}, \text{GuardWidth}, \text{GateConPos}, \text{GateContNum}) + K_4 \\ = K_4 + K_6 \quad (4.6)$$

$$Y_{gr2} = f(\text{MOS}type, \text{GuardConfig}, \text{GuardWidth}, \text{GateConPos}, \text{GateContNum}) + K_4 \\ = K_4 + K_7 \quad (4.7)$$

where the variables in Equations (4.3) to (4.7) are those described in Section 4.1.1. Since X_a is directly proportional to the number of fingers n , Equation (4.3) can be approximated to $X_a \cong nK_1 + K_2$, where K_1 and K_2 are constants that represent the horizontal area occupied by one finger and by the polygons surrounding that finger respectively. The constant K_3 in Equation (4.4) represents the overhang design rule of the gate polygons over the transistor's diffusion region. Similarly, constant K_4 is the separation design rule between the active region and the guard ring. The contribution of the variables in Equations (4.5) to (4.7) to the overall value of X and Y is essentially

constant over the range of interest values for W and n . Thus, the constants K_5 , K_6 , and K_7 are linear combinations of the spacings and sizes design rules of related polygons in the transistor.

The overall width and height can be simplified by collecting the constant terms as shown in Equations (4.8) and (4.9). They only depend on two key parameters: the number of fingers n , and the transistor width W . The constants K'_2 and K'_3 are the combination of constants K_2 to K_7 .

$$X = f(n) = nK_1 + K_2 + 2(K_4 + K_5) = nK_1 + K'_2 \quad (4.8)$$

$$Y = f(n, W) = W/n + K_3 + 2K_4 + K_6 + K_7 = W/n + K'_3 = W_f + K'_3 \quad (4.9)$$

4.1.3 Design Constraints

The design space that limits the feasible solutions for fingered MOS transistors can be determined with a set of process design rules and/or user specified constraints.

- i) The width of each fingered MOS transistor is bounded by user specified minimum and maximum values represented by W_{fmin} and W_{fmax} respectively. The corresponding minimum and maximum height of the transistor are defined by Y_{min} and Y_{max} where:

$$Y_{min} = W_{fmin} + K'_3 \quad (4.10)$$

$$Y_{max} = W_{fmax} + K'_3 \quad (4.11)$$

- ii) For a given L , the user can restrict the total width of the transistor to be within a range of values. This enforces the design space to be in the region bounded by equations Y_1 and Y_2 which describe the layout solutions for transistors having a minimum and maximum width defined by W_{min} and W_{max} respectively.

$$Y_1 = W_{min}/n + K'_3 \quad (4.12)$$

$$Y_2 = W_{max}/n + K'_3 \quad (4.13)$$

4.1.4 Typical Shape Functions and Example Layouts

The shape functions of PMOS and NMOS transistors with increasing W and a fixed value of L , constructed on a $0.8\ \mu\text{m}$ BiCMOS [8] and a $1.2\ \mu\text{m}$ CMOS process [5] are shown in Figures 4.2 and 4.3. The points in the shape functions correspond to solutions of transistors with a number of finger incremented by two, going from left to right. Sample layouts generated for two discrete solutions in both target processes are illustrated from Figures 4.4 to 4.7. The minimum and maximum width, W_{fmin} and W_{fmax} , of each finger in these examples are $5\ \mu\text{m}$ and $250\ \mu\text{m}$ respectively. Furthermore, the minimum and maximum width of the MOSFETs are equal to $100\ \mu\text{m}$ and $1000\ \mu\text{m}$. Note that the NMOS transistors synthesized in the $1.2\ \mu\text{m}$ CMOS process shown in Figure 4.7 do not have a guard ring around them. The key difference found in the shape functions of the two processes are the numerical values of the physical design rules that compose constants K_1 , K_2, K_7 .

(a) $0.8 \mu\text{m}$ BiCMOS process. (b) $1.2 \mu\text{m}$ CMOS processFIGURE 4.2 Shape function of PMOS transistors with $W_{\min} = 5 \mu\text{m}$ and $W_{\max} = 250 \mu\text{m}$.

(a) $0.8 \mu\text{m}$ BiCMOS process. (b) $1.2 \mu\text{m}$ CMOS process.FIGURE 4.3 Shape function of NMOS transistors with $W_{\min} = 5 \mu\text{m}$ and $W_{\max} = 250 \mu\text{m}$

(a) 10-finger with $W=300\ \mu\text{m}$, $W_f=30\ \mu\text{m}$, $X=105.5\ \mu\text{m}$ and $Y=78.5\ \mu\text{m}$.

(b) 8-finger with $W=300\ \mu\text{m}$, $W_f=37.5\ \mu\text{m}$, $X=93\ \mu\text{m}$ and $Y=85.5\ \mu\text{m}$.

FIGURE 4.4 Synthesized layouts of PMOS transistors in a $0.8\ \mu\text{m}$ BiCMOS process.

(a) 10-finger with $W=300\ \mu\text{m}$, $W_f=30\ \mu\text{m}$, $X=106.6\ \mu\text{m}$ and $Y=74.5\ \mu\text{m}$.

(b) 8-finger with $W=300\ \mu\text{m}$, $W_f=37.5\ \mu\text{m}$, $X=89.7\ \mu\text{m}$ and $Y=87\ \mu\text{m}$.

FIGURE 4.5 Synthesized layouts of PMOS transistors in a $1.2\ \mu\text{m}$ CMOS process.

(a) 10-finger with $W=300\ \mu\text{m}$, $W_f=30\ \mu\text{m}$, $X=95.2\ \mu\text{m}$ and $Y=69.3\ \mu\text{m}$.

(b) 8-finger with $W=300\ \mu\text{m}$, $W_f=37.5\ \mu\text{m}$, $X=82.3\ \mu\text{m}$ and $Y=77.4\ \mu\text{m}$.

FIGURE 4.6 Synthesized layouts of NMOS transistors in a $0.8\ \mu\text{m}$ BiCMOS process.

(a) 10-finger with $W=300\text{ }\mu\text{m}$, $W_f=30\text{ }\mu\text{m}$, $X=74.1\text{ }\mu\text{m}$ and $Y=48.7\text{ }\mu\text{m}$.

(b) 8-finger with $W=300\text{ }\mu\text{m}$, $W_f=37.5\text{ }\mu\text{m}$, $X=60.8\text{ }\mu\text{m}$ and $Y=56.3\text{ }\mu\text{m}$.

FIGURE 4.7 Synthesized layouts of NMOS transistors without a guard ring in a $1.2\text{ }\mu\text{m}$ CMOS process.

4.2 Bipolar Transistors

4.2.1 Module Generator

Bipolar transistors are specified by the size of their emitter. This emitter can have any aspect ratio and based on which, the remaining layout polygons in the transistor are constructed and placed.

For the specific layout process used for the prototype implementation (0.8 μm BiCMOS process [8]), the size of the emitter is restricted to be an integer multiple of a reference transistor. This is due to scaling limitations of device simulation models for the prototype layout process [17]. Another limitation of the target process is that only NPN transistors can be constructed. As such, the module generator and shape function for bipolar transistors described in this section are a special case of study.

The reference transistor is designated as 1X and has an emitter area of $W_{\text{Eref}} \times L_{\text{Eref}}$ where W_{Eref} and L_{Eref} are the width and length respectively. Other transistors are designated as 2X, 3X, 4X, etc., each of them has an emitter length, L_E , equal to a multiple of that of the 1X transistor; the emitter's width is kept constant. For example, a 3X transistor has an emitter area of $W_{\text{Eref}} \times 3L_{\text{Eref}}$.

Similar to the case of large MOS transistors, bipolar transistors with large emitter size are constructed by connecting a number of smaller emitters, n_E , in parallel. Each small emitter is referred to as emitter finger and has an area equal to $W_{\text{Eref}} \times L_{\text{Ef}}$ where $L_{\text{Ef}} = L_E / n_E$. As such, there are many combinations of L_{Ef} and n_E for a constant emitter aspect ratio of L_E / W_E . As a result, the shape function of the bipolar transistors is discrete. A continuous shape function can be derived from its discrete counterpart by simply allowing the emitter to take on any aspect ratio. The contacts for the base and the collector in a large bipolar transistor are also split into smaller fingered contact stripes. The number of base and collector contact stripes are designated as n_B and n_C . Furthermore, the collector contact can also be constructed as a ring surrounding the base and the emitter.

The input parameters that the bipolar generator accepts to synthesize corresponding layouts are listed below. The name of the input parameters are emphasized and placed within parenthesis.

1. Type of transistor (*BJTType*): NPN or PNP. Note that only the NPN type is currently supported;
2. The width and length of the emitter (W_E and L_E): these parameters define the size of the transistor which is currently restricted to be a multiple of the 1X transistor;
3. Number of emitter fingers (n_E);
4. Size of emitter fingers (W_{Ef} and L_{Ef}): the width of the emitter finger is currently restricted to be W_{Eref} ;
5. Number of base contact fingers (n_B): the base contact can be formed with smaller contact stripes to satisfy the overall aspect ratio of the cell. The number of base contact stripe is currently set to be one unit larger than n_E ;
6. The width and length of a base contact finger (W_B and L_B): the width is usually fixed and depends on the size of the contacts. The length is currently equal to the length of each emitter finger (L_{Ef});
7. Number of collector contact fingers (n_C): this number can either be 0, 1 or 2. If a full ring collector contact is required, then $n_C = 0$;
8. The width and length of a collector contact finger (W_C and L_C): the width is currently set to a constant due to a specific design rule associated to the NPLUG material used in the target process. The length is the sum of the size of the base polygon and a constant;
9. Collector configuration (*ColRing*): full ring or fingers. If $ColRing = 1$ then a full ring collector contact is constructed otherwise, the number of collector contact fingers is defined by parameter n_C described above;
10. Highest metal layer used to connect to the emitter (*EmitMet*): the highest metal layer constructed to connect to the emitter is mainly defined for routing purposes;
11. Netlist and pin positions (Netlist);
12. Position (*Position*) and orientation (*Orientation*) of the final layout: variable Position is the center coordinates of the layout cell within the floorplan.

4.2.2 Shape Function

Figure 4.8 shows an abstract layout representation of a fingered NPN bipolar transistor.

FIGURE 4.8 Abstract layout representation of fingered bipolar transistor.

The width X of the bipolar is derived as follows:

$$X = X_B + X_{C1} + X_{C2} + 2K_1 \quad (4.14)$$

$$X_B = f(n_E, n_B, W_{Eref}, W_B) = n_E W_{Eref} + n_B W_B + (n_E + n_B - 1)K_2 + K_3 \quad (4.15)$$

$$X_{C1} = f(n_C, ColRing, W_C) = (n_{C1} + 2ColRing)(W_C + K_4) = K_5 \quad (4.16)$$

$$X_{C2} = f(n_C, ColRing, W_C) = (n_{C2} + 2ColRing)(W_C + K_4) = K_6 \quad (4.17)$$

where X_B is the contribution of the emitter and the base contact fingers to form the width of the base polygon. The constant K_1 is the overlap of the n-well collector over the collector contact. The variable W_B in Equation (4.15) is the width of a base contact stripe. Constant K_2 represents the spacing between an emitter and a base contact finger, and constant K_3 is the overlapping rule associated to a base polygon over the outermost base contact and/or emitter fingers. Usually, the number of base contact fingers n_B is assumed to be one unit more than n_E . Replacing $n_B = n_E + 1$ in Equation (4.15) and collecting all constant terms, X_B is reduced to a function of n_E .

$$\begin{aligned}
 X_B = f(n_E) &= n_E W_{Eref} + (n_E + 1)W_B + (n_E + n_E + 1 - 1)K_2 + K_3 \\
 &= n_E(W_{Eref} + W_B + 2K_2) + W_B + K_3 = n_E K_1' + K_2'
 \end{aligned} \quad (4.18)$$

The contribution of the collector to the width of the cell, X_{C1} and X_{C2} , is a function of the number of collector contact fingers (n_C), ring configuration (ColRing) as well as the width of each collector finger (W_C). Constant K_4 is the separation between the base polygon and a collector contact (finger or ring). Variable ColRing is a boolean that reflects whether the collector contact is constructed as a ring configuration or not. Variables n_{C1} and n_{C2} are the number of collector finger on each side of the base polygon where $n_C = n_{C1} + n_{C2}$. If a ring configuration is present, then ColRing = 1 and $n_C = 0$, otherwise ColRing = 0 and $n_C = 1, 2$. Thus, variables X_{C1} and X_{C2} are different only when there is no collector contact ring and the number of collector contact finger is 1. For a specified ColRing and n_C , X_{C1} and X_{C2} can be replaced by two constants, K_5 and K_6 , respectively as W_C is usually constant. The total width X can be simplified by collecting the constant terms. The resulting equation is a function of n_E where K_3' is the sum of the constants in Equations (4.14) to (4.17).

$$X = f(n_E) = n_E K_1' + K_2' + K_5 + K_6 + 2K_1 = n_E K_1' + K_3' \quad (4.19)$$

Similarly, the height Y of a fingered bipolar transistor can be derived as follows:

$$Y = Y_C + 2K_1 \quad (4.20)$$

$$\begin{aligned}
 Y_C = f(n_E, L_E, \text{ColRing}, W_C) &= L_E/n_E + K_7 + \text{ColRing}W_C \\
 &= L_{Ef} + K_6 + \text{ColRing}W_C
 \end{aligned} \quad (4.21)$$

Variable Y_C represents the height of the collector contact (contact finger or contact ring). Constant K_7 is the sum of two components: (i) the overlapping design rule of the base polygon over an emitter finger, and (ii) the separation between the base polygon and the collector contact ring, or the outer edge of a collector finger (refer to Figure 4.8). Given that ColRing and W_C are fixed, Y_C and Y depend only on n_E and L_E .

$$Y_C = f(n_E, L_E) = L_E/n_E + K_4' = L_{Ef} + K_4' \quad (4.22)$$

$$Y = L_E/n_E + K_5' = L_{Ef} + K_5' \quad (4.23)$$

where K_4' is the combination of K_7 and $\text{ColRing}W_C$ and K_5' is the sum of K_4' and $2K_1$.

4.2.3 Design Constraints

The design space of the bipolar transistors is delimited by the following constraints:

- i) The length (i.e. height) of each emitter finger is limited by a minimum and a maximum value designated by L_{Efmin} and L_{Efmax} , where $L_{Efmin} = L_{Eref}$. The corresponding minimum and maximum cell's height is defined by Y_{min} and Y_{max} where:

$$Y_{min} = L_{Efmin} + K'_s \quad (4.24)$$

$$Y_{max} = L_{Efmax} + K'_s \quad (4.25)$$

- ii) The emitter area is bounded by a minimum and a maximum value. This constraint enforces the design space to be in the region bounded by Y_1 and Y_2 , where:

$$Y_1 = L_{Eref}/n_E + K'_s \quad (4.26)$$

$$Y_2 = L_{Emax}/n_E + K'_s \quad (4.27)$$

4.2.4 Typical Shape Function and Example Layouts

Figure 4.9 shows the shape function of NPN transistors with a collector contact ring and a standard 1X transistor having an emitter area equal to $0.8 \times 4.0 \mu m^2$. As can be seen for the case of a 20X transistor, there are six discrete solutions each having the product $n_E L_{Ef} = 20 L_{Eref}$. The minimum and maximum emitter length, L_{Efmin} and L_{Efmax} , imposed on the design space are equal to L_{Eref} and $40 \times L_{Eref}$ respectively. The maximum emitter device in this example also has an emitter length equal to $40 \times L_{Eref}$.

The layout of a 20X transistor for two of the discrete solutions are shown in Figure 4.10. In addition, two sample layouts of NPN transistors without a collector contact ring are illustrated in Figure 4.11.

FIGURE 4.9 Shape function of NPN bipolar transistors in a 0.8 μm BiCMOS process with a collector ring, $L_{Emun}=L_{Eun}=L_{Eref}$, $L_{Efm}=L_{Ema}=40 \times L_{Eref}$.

(a) 20X NPN transistor with 2 emitter fingers, 3 base fingers and a collector ring; emitter connector is metal1; $L_E = 80 \mu\text{m}$, $L_{Ef} = 40 \mu\text{m}$, $L_{Eref} = 4 \mu\text{m}$, $W_{Eref} = 0.8 \mu\text{m}$, $X = 22.4 \mu\text{m}$ and $Y = 57.9 \mu\text{m}$.

(b) 20X NPN transistor with 4 emitter fingers, 5 base fingers and a collector ring; emitter connector is metal1; $L_E = 80 \mu\text{m}$, $L_{Ef} = 20 \mu\text{m}$, $L_{Eref} = 4 \mu\text{m}$, $W_{Eref} = 0.8 \mu\text{m}$, $X = 32.4 \mu\text{m}$ and $Y = 37.8 \mu\text{m}$.

FIGURE 4.10 Synthesized layouts of bipolar transistor in $0.8 \mu\text{m}$ BiCMOS process.

(a) 1X NPN transistor with 1 emitter finger, 2 base contact fingers and 1 collector contact finger; emitter connector is metal2; $L_E = L_{E_{ref}} = 4 \mu\text{m}$, $W_E = 0.8 \mu\text{m}$, $X = 13.9 \mu\text{m}$ and $Y = 8.8 \mu\text{m}$.

(b) 4X NPN transistor with 2 emitter fingers, 3 base contact fingers and 2 collector contact fingers; emitter connector is metal2; $L_E = 16 \mu\text{m}$, $L_{Ef} = 3 \mu\text{m}$, $L_{E_{ref}} = 4 \mu\text{m}$, $W_E = 0.8 \mu\text{m}$, $X = 22.4 \mu\text{m}$ and $Y = 13.2 \mu\text{m}$.

FIGURE 4.11 Synthesized NPN transistors in a $0.8 \mu\text{m}$ BiCMOS process without a collector contact ring.

4.3 Diodes

4.3.1 Module Generator

A diode's electrical characteristics are determined by the dimensions of its active area. For a given area, the width and height of the diode can be varied to provide some flexibility in the aspect ratio of its resulting layout within a floorplan. For a p+/n diode, the active area corresponds to the anode and for a n+/p diode, the active area is the cathode. Figure 4.12 shows an abstract layout representation of a p+/n and a n+/p diode. Special layout considerations are included in the generator to support the layout guidelines described in Sections 2.2.2 and 2.3.2 for ESD protection and latch-up prevention.

FIGURE 4.12 Abstract layout representation of a diode.

The input parameters that the diode generators can accept are summarized as follows:

1. Diode type (*DiodeType*): p+/n or n+/p;
2. Value of active area in μm^2 (A_e);
3. The width and length of the active area (X_a and Y_a);
4. The overlapping values of the polygon other than the active area over the active area (X'_a and Y'_a);
5. Guard ring is required or not (*Guard*): the guard ring is used as a latch-up prevention measure;
6. Width of guard ring surrounding the diode (*GuardWidth*): this parameter depends on latch-up;
7. Contact spacings in the anode/cathode, cathode/anode as well as in the guard ring (*ContSpa*): for protection against ESD, the contact spacings are usually made larger than the minimum value specified by a target design rule;

8. Netlist with pins position (*Netlist*);
9. Layers of the pins (*PinLayer*): if the layer of the wire connecting to the diode is known, the layer of the pins are matched with that of the wires; otherwise, metal is used;
10. Position (*Position*) and orientation (*Orientation*) of the final layout.

4.3.2 Shape Function

Given an equivalent active area $A_e = X_a \times Y_a$, the boundary of a diode can be derived as follows:

$$X = X_a + 2X'_a + 2X_{gr} \quad (4.28)$$

$$Y = Y_a + 2Y'_a + 2Y_{gr} \quad (4.29)$$

$$X'_a = Y'_a = K_1 \quad (4.30)$$

$$X_{gr} = Y_{gr} = f(\text{Guard}, \text{GuardWidth}) = K_2 \quad (4.31)$$

$$\begin{aligned} Y &= Y_a + 2Y'_a + 2Y_{gr} = Y_a + 2X'_a + 2X_{gr} \\ &= A_e/X_a + 2X'_a + 2X_{gr} = A_e/[X - (2X'_a + 2X_{gr})] + 2X'_a + 2X_{gr} \end{aligned} \quad (4.32)$$

where X_a and Y_a are the dimensions of the active area and $X_a, Y_a \neq 0$. The variables X'_a and Y'_a represent the horizontal and vertical distances between the edges of the active area and the outer edges of the polygon of opposite diffusion type from that of the active area. These variables are typically constant for a given layout process. Similarly, the variables X_{gr} and Y_{gr} are the distances between the edges of the diode and the outer edges of the guard ring. These guard ring components are also constant for a given layout process and guard ring width.

The above equations can be further simplified by substituting X'_a and X_{gr} by their respective constant. Upon collecting the constant terms, X and Y are simple functions of A_e and X_a , where K'_1 is the sum of $2K_1$ and $2K_2$.

$$X = f(X_a) = X_a + K'_1 \quad (4.33)$$

$$Y = f(A_e, X_a) = A_e/X_a + K'_1 = A_e/[X - K'_1] + K'_1 \quad (4.34)$$

4.3.3 Design Constraints

The design space for a diode is delimited by the following constraints:

- i) The equivalent active area is limited by a minimum and a maximum value, A_{emin} and A_{emax} , respectively. The height of the diodes with an equivalent active area that satisfies these conditions is bounded by $Y_1 \leq Y \leq Y_2$ where Y_1 and Y_2 are equations that describe the height of the layouts with A_{emin} and A_{emax} , respectively.

$$Y_1 = A_{emin}/(X - K'_1) + K'_1 \quad (4.35)$$

$$Y_2 = A_{emax}/(X - K'_1) + K'_1 \quad (4.36)$$

- ii) The minimum and maximum aspect ratio of the active area is bounded by $\alpha_1 \leq X_a/Y_a \leq \alpha_2$, where $\alpha_1 = 1/\alpha_2$ and $\alpha_1, \alpha_2 \neq 0$. It can easily be shown that the design space must lie between the lines Y_3 and Y_4 , where:

$$Y_3 = [X - K'_1(1 - \alpha_1)]/\alpha_1 \quad (4.37)$$

$$Y_4 = [X - K'_1(1 - \alpha_2)]/\alpha_2 \quad (4.38)$$

4.3.4 Typical Shape Function and Example Layouts

The shape functions of a p+/n and n+/p diode with different equivalent active areas are shown in Figures 4.13 and 4.14, respectively. Examples of discrete implementations generated in a BiCMOS and a CMOS process are illustrated in Figures 4.15 to 4.18. In these examples, the equivalent area is bounded by $A_{emin} = 250 \mu m^2$ and $A_{emax} = 1250 \mu m^2$. In addition, the active area's aspect ratio defined by X_a/Y_a is constrained by $\alpha_1 = 1/5$ and $\alpha_2 = 5$.

 (a) $0.8 \mu\text{m}$ BiCMOS process.

 (b) $1.2 \mu\text{m}$ CMOS process.

FIGURE 4.13 Shape function of p+/n diodes.

 (a) 0.8 μm BiCMOS process.

 (b) 1.2 μm CMOS process.

FIGURE 4.14 Shape function of n+/p diodes.

(a) $A_c = 550 \mu\text{m}^2$, $X_a = 55 \mu\text{m}$, $Y_a = 10 \mu\text{m}$, $X = 88.2 \mu\text{m}$ and $Y = 43.2 \mu\text{m}$.

(b) $A_c = 550 \mu\text{m}^2$, $X_a = 45 \mu\text{m}$, $Y_a = 12.2 \mu\text{m}$, $X = 78.2 \mu\text{m}$ and $Y = 45.4 \mu\text{m}$.

FIGURE 4.15 Synthesized layouts of p+/n diodes in a 0.8 μm BiCMOS process.

(a) $A_c = 550 \mu\text{m}^2$, $X_a = 55 \mu\text{m}$, $Y_a = 10 \mu\text{m}$, $X = 82.3 \mu\text{m}$ and $Y = 39.3 \mu\text{m}$.

(b) $A_c = 550 \mu\text{m}^2$, $X_a = 45 \mu\text{m}$, $Y_a = 12.2 \mu\text{m}$, $X = 72.2 \mu\text{m}$ and $Y = 41.5 \mu\text{m}$.

FIGURE 4.16 Synthesized layouts of p+/n diodes in a 1.2 μm CMOS process.

(a) $A_c \approx 550 \mu\text{m}^2$, $X_a = 55 \mu\text{m}$, $Y_a = 10 \mu\text{m}$, $X = 80.3 \mu\text{m}$ and $Y = 32.8 \mu\text{m}$.

(b) $A_c \approx 550 \mu\text{m}^2$, $X_a = 45 \mu\text{m}$, $Y_a = 12.2 \mu\text{m}$, $X = 70.2 \mu\text{m}$ and $Y = 35 \mu\text{m}$.

FIGURE 4.17 Synthesized layouts of n+/p diodes in a 0.8 μm BiCMOS process.

(a) $A_c = 550 \mu\text{m}^2$, $X_a = 55 \mu\text{m}$, $Y_a = 10 \mu\text{m}$, $X = 86.9 \mu\text{m}$ and $Y = 39.7 \mu\text{m}$.

(b) $A_c = 550 \mu\text{m}^2$, $X_a = 45 \mu\text{m}$, $Y_a = 12.2 \mu\text{m}$, $X = 76.8 \mu\text{m}$ and $Y = 41.7 \mu\text{m}$.

FIGURE 4.18 Synthesized layouts of n+/p diodes in a 1.2 μm CMOS process.

4.4 Rectangular Resistors

4.4.1 Module Generator

The total resistance of a rectangular resistor shown in Figure 4.19 is the sum of four components: (1) the sheet resistance of the contact regions (which can be salicided or not), (2) the resistance due to the number of contacts in the contact regions, (3) the edge resistance between the contact and the resistive regions, and (4) the sheet resistance of the resistive region.

For a resistor with a large resistive region (having 20 squares or more as in the case of the resistor found in I/O pads), the resistance contributed by items (1-3) is negligible with respect to that of the resistive region. Thus, the resistance of a rectangular resistor can be approximated as $R \cong R_s L_r / W_r$ (Ω) where R_s , L_r and W_r are the sheet resistance, length and width of the resistive region, respectively.

(a) Equivalent layout representation.

(b) Equivalent lumped circuit representation.

FIGURE 4.19 Layout and schematic representation of a rectangular resistor.

The input parameters that a rectangular resistor generator can accept are summarized as follows:

1. Resistor value (R);
2. Resistor material (*Material*): the sheet resistance of the resistive region can be determined from this parameter;
3. Length and width the resistive area (L_r and W_r);
4. Number of contacts at each end of the resistor (*TotalCont1* and *TotalCont2*);
5. Netlist (*Netlist*);
6. Layer of pins (*PinLayer*);
7. Position (*Position*) and orientation (*Orientation*) of final layout: the position of the final layout and its orientation within the floorplan;
8. Position (*Index*) of the rectangular resistor within a tiled resistor. As will be discussed in Section 4.5, a tiled resistor is comprised of a set of smaller rectangular resistors being placed in a matrix format, and connecting in a parallel or in a series combination. This parameter is only used when a tiled resistor is constructed.

4.4.2 Shape Function

The equations that describe the shape function of rectangular resistors are derived as follows:

$$W = W_r = W_{c1} = W_{c2} \quad (4.39)$$

$$L = L_r + L_{c1} + L_{c2} = R W_r / R_s + L_{c1} + L_{c2} \quad (4.40)$$

where W_{c1} and W_{c2} , L_{c1} and L_{c2} are the widths and lengths of the contact regions.

Using W_r as the independent variable, L_r , L_{c1} and L_{c2} can be found as follows:

$$\text{ContactsPerCol} = (W_r - K_2) / K_1 \quad (4.41)$$

$$\text{NumCols}_{c1} = \text{TotalCont1} / \text{ContPerCol} \quad (4.42)$$

$$\text{NumCols}_{c2} = \text{TotalCont2} / \text{ContPerCol} \quad (4.43)$$

$$L_{c1} = \text{NumCols}_{c1} K_3 + K_4 \quad (4.44)$$

$$L_{c2} = \text{NumCols}_{c2} K_3 + K_4 \quad (4.45)$$

The number of contacts per column, ContPerCol , in Equation (4.41) is obtained by dividing the difference between the width of the resistor and K_2 by the vertical spacing that one contact occupies. This vertical spacing is represented by K_1 . The constant K_2 is

the total overlap from the edges of the resistor to the nearest contacts, measured in the direction of the resistor width. The variables TotalCont1 and TotalCont2 are user specified parameters and represent the number of contacts to be placed at the two ends of a resistor. From these parameters, the number of contact columns required for each contact section can be obtained with Equations (4.42) and (4.43). The length of the contact regions, L_{c1} and L_{c2} , are determined with Equations (4.44) and (4.45). The constant K_3 in these equations is the space occupied by one contact column. Finally, constant K_4 is the sum of two components: (1) the separation between the innermost contacts and an edge of the resistive region, and (2) the separation between the outermost contacts and the outer edge of the resistor.

The equation that characterizes the length of the resistor can be simplified by substituting Equations (4.41) to (4.45) into Equation (4.40) and collecting the constant terms. The final equation of L is a function of W_r and R where:

$$L = f(W_r, R) = RW_r/R_s + K'_1/(W_r - K_2) + K'_2 \quad (4.46)$$

$$K'_1 = (TotalCont1 + TotalCont2) K_1 K_3 \quad (4.47)$$

$$K'_2 = 2K_4 \quad (4.48)$$

4.4.3 Design Constraints

Three constraints are used to limit a resistor's design space and are described below:

- i) Minimum width W_{min} -- for a maximum operating current I_{max} and a maximum current density J_{max} (mA/ μ m) of a given resistive layer, the minimum width W_{min} of a resistor is $W_{min} = I_{max}/J_{max}$. Since W_{min} is typically smaller (few microns) than the area occupied by each column of contacts (high ten's of microns), W_{min} is usually dictated by the number of contacts required in each contact area as will be described in (ii);
- ii) minimum number of contacts in each contact region -- typically, the contact region that connects the resistor to the pad opening requires more contacts than that connecting the resistor to the output drivers or to the input buffer;

- iii) maximum width $W_{\max}$ -- this limit is usually a user specified parameter. Some considerations should be made in the selection of W . In the case of long resistors (many squares), increasing the width will increase the accuracy of the resistor value, but the overall area and parasitic capacitance will also increase. In addition, the parasitic diode used as ESD protection device for abnormal negative transients results in a larger junction voltage as W increases. Furthermore, increasing the width of a short resistor also improves the accuracy because the edge resistance between the contact and the resistive regions decreases.

4.4.4 Typical Shape Function and Example Layouts

A typical shape function of rectangular resistors is shown in Figure 4.20. Example layouts for a 1 k Ω n-well resistor constructed for a BiCMOS and a 1 k Ω n-device well resistor CMOS process are illustrated in Figures 4.21 and 4.22. The sheet resistance for these design processes are 50 Ω /square and 45 Ω /square respectively. In these examples, the design space restricts the width of the resistors to be within $W_{\min} = 12.5 \mu\text{m}$ and $W_{\max} = 37.5 \mu\text{m}$. In addition, a minimum of 5 and 15 contacts at each end of the resistor are required with the end connecting to the pad opening having a larger number of contacts.

(a) 0.8 μm BiCMOS process. (b) 1.2 μm CMOS process.

FIGURE 4.20 Shape functions of rectangular resistors.

 FIGURE 4.21 Synthesized layouts for n-well resistors in a 0.8 μm BiCMOS process.

 FIGURE 4.22 Synthesized layouts for n-device well resistors in a 1.2 μm CMOS process.

4.5 Tiled Resistors

4.5.1 Module Generator

One of the most difficult device to manufacture within tight tolerances is the resistor. The difference between the extracted resistance and the actual resistance within the wafer may be greater than 10% [17]. The manufacturing error is an important issue for circuits where the ratio of two resistor values is critical to ECL circuits. One way to remedy this problem is to construct each resistor as a parallel combination of smaller resistor tiles each having a resistance value of R_t . The relationship between the total resistance R with n resistor tiles connected in parallel is shown in Equation (4.50). For n resistor tiles, there are n_r rows and n_c columns where:

$$n = n_r n_c \quad (4.49)$$

$$R = R_t / n = R_t / n_r n_c \quad (4.50)$$

FIGURE 4.23 Abstract layout representation of tiled resistors connected in parallel with $n_r = 2$ and $n_c = 3$.

The type of parameters that a tile resistor generator accepts are as follows:

1. Total resistance value (R);
2. Either the value of each resistor tile (R_t) or total number of tiles (n_t);
3. Either the number of tiles rows (n_r) or columns (n_c);
4. Resistor material (*Material*): the sheet resistance of the resistive region can be determined from this parameter;

5. The size of each resistor tile: either the length or width (L_t or W_t);
6. Tile resistor configuration (*Config*): parallel or series. Note that only the parallel configuration is currently supported. The key difference between the two configurations is in the way the wiring among the tiles is done;
7. Number of total contacts at each end of each resistor tile (*TotalCont1* and *TotalCont2*);
8. Netlist (*Netlist*);
9. Layer of pins (*PinLayer*);
10. Position (*Position*) and orientation (*Orientation*): the position and orientation of the final layout within a floorplan.

4.5.2 Shape Function

The width and height, X and Y , of a tiled resistor connecting in parallel can be derived as follows:

$$X = f(n_c, W_t) = n_c W_t + (n_c - 1) K_1 + K_2 \quad (4.51)$$

$$Y = f(n_r, L_t) = n_r L_t + (n_r - 1) K_3 \quad (4.52)$$

where W_t and L_t are the width and length of a rectangular resistor tile. Constants K_1 and K_3 represent the separation between two columns and two rows respectively, and constant K_2 is the area occupied by the wires that connect the rows together.

For selected values of R and R_t , Equations (4.51) and (4.52) can be simplified to functions which only depend on n_c and W_t . This is achieved by substituting $n_r = R_t/n_c R$ and $L_t = R_t W_t / R_s$, and by collecting the constant terms.

$$X = f(n_c, W_t) = n_c (W_t + K_1) + K'_1 \quad (4.53)$$

$$Y = f(n_c, W_t) = (R_t/n_c R) [(R_t W_t / R_s) + K_3] + K_3 \quad (4.54)$$

where $K'_1 = K_2 + K_1$ and R_s is sheet resistance of the resistive region. Since $n = n_r n_c$, the shape function of a tiled resistor is composed of a set of curves, each of which corresponds to a combination of n_r and n_c with varying resistor tile's width and length.

4.5.3 Design Constraints

The design space of a tiled resistor can be reduced by a set of constraints which can be summarized as follows:

- i) Minimum and maximum sizes for the resistor tile for a given value of R_t . These constraints are the same as those described in Section 4.4 where the widths of the resistor tile are limited within a range of values.
- ii) The aspect ratio of the final layout can be restricted to be within two limits bounded by α_1 and α_2 .

$$\alpha_1 \leq X/Y \leq \alpha_2 \quad (4.55)$$

4.5.4 Typical Shape Function and Example Layouts

Figure 4.24 shows two graphs that represent the shape function of a 200 Ω polysilicon tiled resistor constructed for a 0.8 μm BiCMOS process. The first plot contains the layout solutions for resistor tiles, R_t , equal to 1.6 $\text{k}\Omega$ while the second one summarizes the solutions for resistor tiles of 4 $\text{k}\Omega$ each. Clearly when $R_t = 1.6 \text{ k}\Omega$, there are four possible combinations of $n = n_r n_c = 8$. Similarly for $R_t = 4 \text{ k}\Omega$, there are six possible choices of $n = n_r n_c = 20$.

Two example layouts are shown in Figure 4.25 where the first layout is a 2x4 resistor for the case with $R_t = 1.6 \text{ k}\Omega$ and the second is a 2x10 combination with each resistor tile $R_t = 4 \text{ k}\Omega$. In both cases, the width of each resistor tile is selected to be 10 μm . The aspect ratio of the final layout is restricted to be within $\alpha_1 = 1/5$ and $\alpha_2 = 5$. In addition, the minimum and maximum allowable width for each tile is 5 μm and 30 μm respectively.

(a) $n = n_r n_c = 8$ tiles, $R_t = 1.6 \text{ k}\Omega$, $\alpha_1 = 1/5$ and $\alpha_2 = 5$.

(b) $n = n_r n_c = 20$ tiles, $R_t = 4 \text{ k}\Omega$, $\alpha_1 = 1/5$ and $\alpha_2 = 5$.

FIGURE 4.24 Shape functions for a 200 Ω polysilicon tiled resistor connected in parallel.

(a) 2x4 parallel configuration with $R = 200 \Omega$, $R_s = 750 \Omega/\text{sq}$, $R_t = 1.6 \text{ k}\Omega$, $L_r = 21.3 \mu\text{m}$, $W_r = 10 \mu\text{m}$, $X = 52.8 \mu\text{m}$ and $Y = 52.3 \mu\text{m}$.

(b) 2x10 parallel configuration with $R = 200 \Omega$, $R_s = 750 \Omega/\text{sq}$, $R_t = 4 \text{ k}\Omega$, $L_r = 53.3 \mu\text{m}$, $W_r = 10 \mu\text{m}$, $X = 122.6 \mu\text{m}$ and $Y = 117.8 \mu\text{m}$.

FIGURE 4.25 Synthesized layouts of tiled resistors in a $0.8 \mu\text{m}$ BiCMOS process.

4.6 Pad Opening

Layout geometries associated with the pad opening are also generated by a special purpose generator. Special rules exist in most processes that dictate the presence or absence of specific design layers under the pad. These rules are required to level the passivation region with the layers of surrounding polygons. In addition, the pad opening is designed with the layer of its connectors matching the layer of the components to which it will be connected. The left and right connectors of the sample layouts in Figure 4.26 are constructed with metal1 as connecting layer while the top and bottom connectors are synthesized for metal2 terminals. Typically, the size of the passivation region depends on the technology of the bonding wire used. The layout geometries are also very standard across most pad macros. For these reasons, no shape function has been derived for the pad opening. Key parameters that distinguish the pad openings include the size of the passivation region, the number of connectors and associate connecting metal layers.

The following list summarizes the input parameters that a pad opening generator can accept:

1. Size of the passivation opening (*PadSize*): this parameter usually depends on the technology of the bonding wire used;
2. Number of connectors and their orientation (*ConnectorInfo*): a pad connector can point to the north, south, east and west;
3. Netlist (*Netlist*);
4. Pin layer (*PinLayer*): the pin layer or connector layer is defined to match the layer of the circuit element that a given pad connects to.
5. Position (*Position*) and orientation (*Orientation*): the center coordinates and orientation of the final layout.

(a) 0.8 μm BiCMOS process: Passivation region is $90 \mu\text{m} \times 90 \mu\text{m}$ and $X = Y = 171.7 \mu\text{m}$.

(b) 1.2 μm CMOS process: Passivation region is $164 \mu\text{m} \times 164 \mu\text{m}$ and $X = Y = 260 \mu\text{m}$.

FIGURE 4.26 Synthesized layouts of a four-connector pad opening.

4.7 Lateral Silicon Controlled Rectifier (LSCR)

The layout of a Lateral Silicon Controlled Rectifier (LSCR) is a lateral p-n-p-n diffusion structure, laid out in two pairs of p-n polygons. The first p-n pair is tied to a V_{ss} power rail. The second pair is butt-connected, laid within an n-well, and tied to the pad opening to collect excess parasitic charges as prevention measure from electrostatic discharge (ESD). As in the case of the pad opening, no shape function has been derived for this circuit element due to standard geometries in most pad designs. Figure 4.27 shows example layouts of LSCRs generated by our module generators for both BiCMOS and CMOS processes.

The parameters that the LSCR generator accepts can be summarized as follows:

1. Size of the four pnpn polygons (*DifSizes*);
2. Spacing between the pn and the but-pn polygons (*DifSpa*);
3. Contacts spacings for the pnpn polygons (*ContSpa*);
4. Netlist (*Netlist*);
5. Position (*Position*) and orientation (*Orientation*) of the final layout.

(a) 0.8 μm BiCMOS process: $X=59.2 \mu\text{m}$ and $Y=37.5 \mu\text{m}$.

(b) 1.2 μm CMOS process: $X=158.6 \mu\text{m}$ and $Y=62.6 \mu\text{m}$.

FIGURE 4.27 Synthesized layout of a LSCR.

4.8 Snap-Back Transistor Generator

The snap-back transistor in the ESD protection circuit is a special case of a NMOS transistor. It is usually constructed as a two-finger NMOS device. For this reason, no special generator has been developed for it. Figure 4.28 show the layouts of a snap-back transistor synthesized with a the help of the MOSFET generator.

(a) 0.8 μm BiCMOS process with $X=22.8\ \mu\text{m}$ and $Y=20.8\ \mu\text{m}$.

(b) 1.2 μm CMOS process with $X = 35.5\ \mu\text{m}$ and $Y = 43.6\ \mu\text{m}$.

FIGURE 4.28 Synthesized layout for a snap-back transistor with $W_f = 15\ \mu\text{m}$.

4.9 Summary

Module generators and shape functions for flexible building blocks in I/O pad macros have been derived. These blocks include the transistors in the output driver: (MOS and bipolar transistors), diodes, resistors and tiled resistors. The layout of other blocks such as the pad opening and the LSCR are standard in most pad macros and as a result, no shape function has been derived for them. The design space of a shape function can be restricted by a number of constraints. Typical constraints are associated with technology limitations or user specified conditions.

Chapter 5

Prototype Implementation

This chapter presents the prototype implementation for the shape functions and corresponding module generators. The emphasis is on the prototype architecture and the software concepts applied for the implementation. The implementation strategy must address the ability to control specific layout features, the flexibility to accommodate layout variances from different processes and the capability to interface with a floorplan, router and compactor.

5.1 Design Concepts for the Synthesis Problem

This section presents the design concepts applied in the context of developing module generators for I/O pad macros to support all the design constraints, considerations for latch-up and ESD as presented in Sections 2.2 and 2.3. The methodology is motivated by two key objectives. The first objective is an implementation scheme that supports process independence and the second is a flexible design architecture that could easily be extended to other circuit structures such as boundary scan circuits and/or sophisticated ESD protection devices. These objectives are best achieved by the concepts of (i) abstraction, (ii) modularity, and (iii) hierarchical architecture. To better appreciate these design fundamentals, it is useful to examine the attributes of the programming language selected for the prototype implementation.

5.2 Language of Construct

The fundamental syntax for the prototype implementation is Common Lisp [3, 24]. It is chosen for a number of reasons. First, Lisp is an object oriented language. It provides a natural means for representing objects, the relationships between them, and most importantly, the property of inheritance which is the key attribute that supports abstraction techniques. Modular design is not a problem for Lisp as it allows procedural calls.

Interactive interface, among other features, is useful for quickly assessing the impact of different routines. Such capability facilitates software maintenance and promotes opportunities for creating high standard programs by providing the designer an ability to exercise incremental, but thorough test cases. Furthermore, Lisp frees the designer of the tedious tasks of type definition and memory management, hence contributing to faster code generation cycles. Finally, Lisp is selected for compatibility with existing software to be used in the synthesis process, namely the floorplan [27] and the router [12].

To standardize a programming style across various components of the synthesis system, a coding convention for Lisp implementation is proposed. The objective is to facilitate software support and readability. The adopted coding convention is described in Appendix A.

5.3 Abstraction

In order to achieve process independence and to obtain a flexible architecture, a sufficient high level of abstraction must be applied in the synthesis process to decouple it from the knowledge of internal layout structures as well as process parameters. In addition, the devised algorithms must be implemented in their most general form as possible to facilitate their adaptation to future technologies. We associate abstraction techniques to three categories: *data abstraction* [19], *procedural abstraction* and *variable abstraction* [15].

Data abstraction is a technique of isolating the parts of a program that deal with how data objects are represented from those that deal with how data objects are used. The representation scheme used in our system for data abstraction makes use of Lisp primitives where each layout object is represented by a *flavor*, for which *instance variables* are defined to distinguish different objects and *methods* are implemented for data manipulation.

Procedural and variable abstractions, on the other hand, deal with abstraction of design rules and user specifications imposed on the layout elements. These specifications are

classified into two categories: quantitative and qualitative specifications. Examples of quantitative specifications include the size constraints for individual features and the numerical values of physical design rules. Qualitative specifications bind specific layer names and layer types to given layout features. For example, in ProcessA, a guard ring may be realized with a device well which must be appropriately doped, whereas in ProcessB, a deep trench may be available for that purpose. Other qualitative information includes latch-up design rules, the nature of the substrate and wells, available interconnect layers and contacting rules. For each *quantitative* specification, there is a corresponding process independent *variable*. Similarly for each *qualitative* specification, there is an equivalent process independent *procedure variable* which is used to instantiate certain layout features. Thus, variable and procedural abstractions are associated with quantitative and qualitative features of a target process respectively. The synthesis system uses the quantitative variables and qualitative procedure variables to construct mask polygons with little knowledge about the process. Any minimum value specified in the process design rules or default layout features that associate with a technology can be overwritten by a user control file without modifying the generators.

5.3.1 Data Abstraction

Designing an effective data representation for the various flavors is a challenge. At the initial stage of the synthesis process, the information required for a flavor instance to describe a circuit component may only contain the aspect ratio or simple electrical properties. The final description, however, contains mainly physical information of numerous layout polygons from which the initial information can be implied. Each synthesis step applies transformations which introduce new physical information to the flavor instance. The information that a flavor carries are the combination of the following attributes: (i) physical features, (ii) electrical properties, (iii) pins for connectivity, and (iv) specific constraints imposed on the layout object to meet latch-up and ESD protection requirements. Thus, the strategy taken to define an abstract flavor is a two step process. First, the attributes of the object are identified. Second, the flavor is devised such that the use of inheritance is maximized to promote code reusability.

5.3.2 Example of Data Abstraction

The flavors that represent the specific layout objects in a I/O cell range from the most simple objects such as contacts, rectangular polygon used as a wire or as a substrate well, to more complex building blocks such as guard ring, pad opening, diodes, transistors, etc. For the purpose of demonstrating data abstraction, the design steps used in the definition of a flavor that represents a guard ring is presented here as a case study. First, the attributes of the guard ring are identified. A guard ring, when viewed from a physical stand point, is an object with four rectangular polygons connected at 90 degrees angles. From the electrical and connectivity point of view, a guard ring is characterized by its layer type, the width of its polygons, the potential to which it is connected as well as the nature of contacts to be placed on it. The width of the polygons, the number of contacts, their sizes and spacings, are among other features, some of the key parameters that dictate how effectively the guard ring will provide protection against latch-up.

Figure 5.1 illustrates the fragmented Lisp codes of three example flavors. The first flavor, termed as *tBox*, represents a rectangular object. The prefix *t* denotes a type -- flavor. The second flavor, called *tRing*, describes an object having the physical characteristics of a ring. This flavor inherits the properties defined by *tBox* to describe its overall aspect ratio. The guard ring described above is a special case of *tRing*, hence the flavor designed for the guard ring, *tGuardRing*, is defined. This flavor inherits the attributes of *tRing* and consequently those of *tBox*. Individual guard rings required for different layout structures become distinct *instances* of *tGuardRing* with a unique binding of the list of attributes in *tGuardRing*. Thus, data abstraction is a powerful method to describe objects with similar attributes but different attribute/value bindings.

2 of/de 2

```

.. Geometry definition of a "rectangular box" Any object with rectangular shape can inherit this flavor to describe
.. the overall dimensions, position and orientation

```

```

(DEFFLAVOR tBox

```

```

  (Width      . Polygon width
   Length     . Polygon length
   (ICenter   '(0 0)) . Polygon center position - initialized
   (IDirection '(1 0)) . Polygon direction vector - initialized
) () :SETTABLE-INSTANCE-VARIABLES)

```

(a) Flavor of a rectangular polygon.

```

.. Geometry and composition of an object with "ring" characteristic. In essence, a ring consists of four rectangular
.. polygons of a given layer type and are connected at 90 degrees angles

```

```

.. - The following attributes must be initialized when a ring is constructed
..   :RingLayer, InnerLength, InnerWidth and StripWidth
..
.. - The following attributes can be initialized but are optional
..   :ICenter and IDirection
..
.. - The following attributes are updated when this ring instance is constructed
..   Length (this is the outer length), Width (this is the outer width), ICenter, IDirection (optional),
..   and IElement

```

```

(DEFFLAVOR tRing

```

```

  (RingLayer . The layer type of this ring
   InnerLength . The inner length of the ring
   InnerWidth . The inner width of the ring
   StripWidth . The width of the strips of this ring
   IElement) . The four strip's instances of this ring
  (tBOX) . The outer length, outer width, center and direction of the ring
          . inherits physical attributes of a rectangular polygon
  :SETTABLE-INSTANCE-VARIABLES)

```

(b) Flavor of a ring.

```

.. Geometry and composition of a "guard ring" The guard ring is a special case of a ring with additional attributes to
.. define its connectivity and requirements for latch-up protection

```

```

.. - The following attributes must be initialized when a guard ring is constructed
..   RingLayer, InnerLength, InnerWidth, StripWidth, IOPin and ILatchUpInfo
..
.. - The following attributes can be initialized but are optional
..   :ICenter and IDirection
..
.. - The following attributes are updated when this guard ring instance is constructed
..   Length (this is the outer length), Width (this is the outer width), ICenter, IDirection (optional),
..   and IElement

```

```

(DEFFLAVOR tGuardRing

```

```

  (IOPin . The netlist node to which the guard ring is connected
   ILatchUpInfo) . List containing the information for latch-up protection
  (tRing) . Inherits the physical and electrical attributes of a ring
  :SETTABLE-INSTANCE-VARIABLES)

```

(c) Flavor of a guard ring.

FIGURE 5.1 Fragmented Lisp codes describing examples of flavors.

5.3.3 Procedural Abstraction

Procedural¹ abstraction is an important technique used to decouple the knowledge of qualitative features from a target process that impose on a layout object. First, it *decomposes* a task into independent *abstract* parts that accomplish identifiable sub-tasks which can be used in sequence to define other tasks. Second, it applies the concept of *procedure binding* where the abstract tasks are represented by *procedure variables*, for which the actual execution is implemented by *target procedures*. At initialization, the procedure variables are bound with target procedures associated with a specific process. The execution of an abstract task must be invoked via procedure variable instead of a call to the target itself. This is to ensure that future modifications to the task is done at one place -- the target procedure. In addition, associated to each procedure variable is a *default target*. The purpose of the default target is to record the fact that a procedure variable does not have a valid target, hence minimizing hidden programming errors. The discussion in the next section describes an example to illustrate how the idea of procedural abstraction is applied in the design of a module generator, and how procedure binding is implemented.

5.3.4 Example of Procedural Abstraction

Usually, many levels of procedural abstraction (decomposition) are necessary to achieve process independence. At the highest level, the task is stated in broad terms such as *construct a p+/n diode layout*. At lower levels, a more procedural orientation is taken. For example, the action of constructing a p+/n diode may imply a long sequence of procedural steps such as (1) construct the diffusion polygon in the diodes's active area, (2) place contacts on this diffusion section, (3) cover the contacts with metal1, (4) determine the geometry of the guard ring surrounding the active area, etc. At the next level of abstraction, these steps can be broken into another set of procedural sequences which, however, require more knowledge about the attributes of the layout object than in the previous stage. In the particular case of placing contacts on the diode's active area, the steps involved may consist of (1) determining the mask layer for the diffusion polygon in

1. All references to procedures in this discussion imply a reference to functions or methods.

the diode's active area, (2) determining the contacts spacings which can be user specified or simply minimum design rules, (3) computing the number of contacts in the active area, etc. These steps need more explicit information about the type of diode, the type of contact used (i.e. contact, via or topvia), as well as the intricate features associated with various layers. For example, the task associated with the determination of mask layer for the diode's active area is an atomic function which takes the diode type as input and returns p+ doping if a p+/n diode is to be constructed, otherwise it returns the n+ doping layer. This simple association may be more complex in one process than in another. For instance, the mask layers of a p+ doping polygon in ProcessA may be a combination of two layers -- a n+ doping exclusion and a p+ doping layer. On the other hand, there may be only one mask layer for ProcessB. This means the function that determines the mask layer must also know specific process information of the target technology.

Figure 5.2 shows fragmented Lisp pseudo-code that illustrates how the task of constructing the diffusion polygon in a diode's active area is achieved with procedural abstraction. The procedure `fConstruct_Diode_DifActiveArea`, in this example, is the highest level of abstraction and is composed of three procedures. They are `fReturn_ActiveAreaLayer`, `fvDetermine_MaskLayer` and `fConstruct_Polygon`. Since `fvDetermine_MaskLayer` needs to know the specific process information, it is a procedure variable -- denoted by the prefix `fv` for function variable. Procedures `fProcessA_Determine_MaskLayer` and `fProcessB_Determine_MaskLayer` are two example targets and are bound to `fvDetermine_MaskLayer` during the initialization stage by `fBind_Diode_ProcVars`. Procedure `fConstruct_Polygon` instantiates rectangles for the given mask layers with specified dimensions. It applies the same operation as many times as there are mask layers found in the list returned by `fvDetermine_MaskLayer`. Thus, process independence can be achieved with procedural abstraction.

```

DEFUN fConstruct_Diode_DifActiveArea (DiodeType, IGeometry)
.....
  AbstractLayer = fReturn_ActiveAreaLayer (DiodeType);
  IMaskLayers = fvDetermine_MaskLayer (AbstractLayer);
  fConstruct_Polygon (IMaskLayer, IGeometry);
END,

```

(a) First level of abstraction.

```

DEFUN fReturn_ActiveAreaLayer (DiodeType)
.....
  IF DiodeType = p+
    THEN RETURN p+;
    ELSE RETURN n+;
END,

DEFUN fConstruct_Polygon (IMaskLayer, IGeometry)
.....
  DO Until IMaskLayer = NIL
    fDraw_CIFBox (IMaskLayer, IGeometry);
END,

```

(b) Second and third level of abstraction.

```

DEFUN fProcessA_Determine_MaskLayer (AbstractLayer)
.....
  CASE AbstractLayer
    p+ RETURN (LIST n+Exclusion, p+Doping);
END,

DEFUN fProcessB_Determine_MaskLayer (AbstractLayer)
.....
  CASE AbstractLayer
    p+ RETURN p+Doping;
END,

DEFUN fBind_Diode_Targets (ProcessType)
.....
  CASE ProcessType
    ProcessA:   fvProcVar1 = fProcessA_Target1;
               fvProcVar2 = fProcessA_Target2;

               fvDetermine_MaskLayer = fProcessA_Determine_MaskLayer;
    ProcessB:   fvProcVar1 = fProcessB_Target1;

               fvDetermine_MaskLayer = fProcessB_Determine_MaskLayer;
END,

```

(c) Target implementations and procedure binding.

FIGURE 5.2 Construction of diode diffusion polygon with procedural abstraction.

5.3.5 Variable Abstraction

Variable abstraction is concerned with the representation of numerical values of physical design rules and physical layout features associated with a given circuit element. The design rules that need abstraction include feature dimensions, spacings and overlappings. An example of physical layout feature is the minimum distance between two large conducting devices below which a guard band is necessary for latch-up prevention. Since this value may differ from one process to another, it is a candidate for abstraction. Variables abstraction is implemented with global variables. Their binding is similar to that of procedure binding. That is, the numerical values associated with a target process are assigned to their global variables during the initialization of the software.

5.4 Modular and Hierarchical Architecture

The concept of modularity consists of dividing a software program into a number of software modules for the purpose of managing complex software. Hierarchical structure merges software components of similar functionality and complexity to a same level of refinement to facilitate software reusability. A typical software module, as shown in Figure 5.3, is composed of three types of sections (files), namely an interface section, an entry section (special case of implementation section), and one or more implementation sections. In addition, each module may contain several definition sections as well as utility files. The interface section contains procedures and variables that may be used by other software components (i.e. allowing external access). The entry section initializes the process independent variables and binds procedure variables with target procedures. In contrast to the interface section, the implementation sections are private from the view of the external software entities. This is to ensure software portability across different design technologies and design releases.

FIGURE 5.3 A typical software module.

5.5 Module Generator Hierarchy

Figure 5.4 shows the organization of the module generators. It is a top-down architecture where each module generator can be added, removed or modified without affecting its neighbors. The generator system reads a process file and a user specifications file to produce an output file containing mask information of the generated circuit elements. The process description file contains quantitative information concerning the numerical values of physical dimensions, spacings, and overlaps, as well as qualitative description of the nature of the various layers. For example, the quantitative section is used to describe the geometries allowed on each layer or specific layout feature, such as 45 degree bevelling at each corner of a pad opening. Qualitative description might include information regarding the presence or absence of a guard ring around a given type of transistor. The user specification file holds electrical properties of each circuit component, a netlist description as well as information of specific layout features.

FIGURE 5.4 Module generators hierarchy.

5.6 Module Generator Program Flow

Figure 5.5 describes the program flow of a typical module generator. There are five steps involved in the synthesis of a layout instance. They are: (i) parameter mapping and parsing, (ii) geometry computation, (iii) translation and/or rotation, (iv) pin alignment to routing grid, and (v) generation of CIF or EDIF code.

5.6.1 Parameter Mapping and Parsing

The parameter mapping and parsing stage can be summarized as follows. First, the parameters of the cell to be synthesized is read from a file. These parameters are mainly user-specified features associated with electrical characteristics (functional, ESD and latch-up), layout configuration, connectivity, pin positions as well as cell position and orientation. A description of these parameters can be found in Chapter 4. The input parameters are written into a flavor instance representing the cell. A check follows to ensure that the specified parameters are valid (i.e. within certain range of values). Warnings are reported in a log file if any inconsistent value is found. If a parameter is *symbolically* (i.e. abstracted) specified as default, then the actual value is mapped to it. All global parameters associated to default values are kept in a single file for maintenance purposes.

5.6.2 Cell Geometry Computation

Cell geometry computation is the main step of the module generator. It determines the sizes, position and orientation for every polygon instance in the layout. Typically, this problem is composed of three stages. The first stage constructs the active parts of the cell and usually involves objects with layers such as diffusions, polysilicons or wells (front end information). The next step deals with interconnect placement. The layers that this stage manipulates are mainly contact, via or topvia as well as metals (back end information). Since some processes do not allow stacked vias as discussed in Section 3.2, the geometry computation of the first two stages may be inter-dependent. The last stage is internal wiring and deals with metal wires. No diffusion nor polysilicon wires are used in the module generators.

FIGURE 5.5 Module generator program flow.

5.6.3 Translation and Rotation

Translation is achieved by a simple routine that operates on the center of the polygons. This routine is composed of two steps. The first step determines the actual values to be added or subtracted from the center of each polygon. The computed values are then used in the next step which is essentially a loop performing an addition operation on every object instances without knowing the physical features of the objects. This is possible because flavors are used and inheritance is applied across the prototype software. The steps involved to rotate a layout cell are the same as for a translation with one exception -- the operation is performed on the unit vector describing the angle of the object with respect to a reference axis $Y = 0$.

5.6.4 Pin Alignment to Routing Grid

As described in Section 3.4, the pins on a layout module must be aligned to a symbolic grid for routing purposes. This process consists of three steps. The first step determines if a pin is already on a grid intersection. If it is not then the next step computes the size of the wire to be added to the pin. The last step consists of creating a wire instance and adds it to the list of elements of the layout object.

5.6.5 Generation of CIF or EDIF Code

A rectangular polygon can be completely specified with three elements: (i) length and width (geometry), (ii) center (position), and (iii) orientation unit vector. Non-rectangular objects can be described in a similar fashion with the exception that the length and width are replaced with a set of coordinates that define the corners of the object. As such, only two routines are required to describe the geometry, position and orientation of a polygon written in CIF or EDIF format: one for rectangular one for non-rectangular objects. Similar to the translation and rotation stage, CIF or EDIF generation is carried out in a loop that extracts the geometrical information from the polygons' instances without knowing the intricate information of the objects. The resulting CIF or EDIF code are written into a file which commercial layout tools can in turn translate into their own format for mask data generation.

5.7 Module Generators Run-Time and Memory Requirements

The run-time of the module generator is $O(n)$ complexity where n is the number of polygons in the layout. Similarly, the space required is also of order $O(n)$ -- one flavor instance for each polygon. Experimental results show that it takes less than one minute to synthesize 20 cells each having an average of several hundred polygons.

5.8 Shape Function Architecture

A typical shape function software module is composed of three interface routines. The first interface is designed to generate data for a floorplanning algorithm as well as for the module generator of a given cell. Relevant data that a floorplan algorithm uses are the dimensions of a cell instance. The other data are mainly useful for the module generators. More specifically, they are being back-annotated by a private routine within the shape function to produce relevant input parameters for the module generator. The other two interface routines are a derivative of the first one. They are defined to return the vertical dimension of a feasible solution given that the horizontal dimension is known, and vice versa. The design constraints associated to each shape function can also be used to limit the set of feasible solutions.

5.9 Shape Function Program Flow

The program flow of a shape function generation is very similar to that of a module generator, consisting of three key stages: (i) parameter parsing, (ii) area modelling, and (iii) dumping of data. Parameter parsing ensures the input parameters are within a range of operating values. Area modelling uses the equations derived for each layout cell in Chapter 4 in conjunction with user parameters and process design rules to determine the length and height of a layout instance. For each solution, it is checked against the design constraints imposed on the cell and is discarded if it is out of bound. The last step writes the feasible solutions to a file which a floorplan and module generators can use for their own purpose.

5.10 Shape Function Run-Time and Memory Requirements

Both run-time and memory requirement for a typical shape function generator is order $O(n)$. Variable n is the number of points that a floorplan algorithm has to explore if no design constraints were to impose on the cell. In the context of I/O pad design, the cell which has the most points in a shape function is the tiled resistor. Experimental results show that the time required to generate a shape function's data is in the order of tens of seconds.

5.11 Experimental Coding Results

The prototype implementation for the module generators and shape function show that approximately 30% of the source code requires duplication and modification to accommodate a new set of design rules. These codes are target variables, functions and methods which must know the intricate details of a layout process. For example, target functions and methods must be created for processes with different type of interconnect rules. In addition, a new set of target global variables must be defined. As far as how a device is synthesized from a high level view is concerned, no modification is required as abstraction hides the target variables from the actual variables.

5.12 Summary

Key concepts applied in the design of shape functions and module generators include abstraction, modularity and hierarchical structure. These concepts support the requirements of controlling specific layout features and provide a means to accommodate the layout variances from different processes as well as to interface with other software components

Application of abstraction as well as object-oriented paradigm provides a framework upon which design portability and code-reuse can be maximized. Properly structured system transforms a complex problem (e.g. manipulation of layout variances) into modular issues which could be solved individually. The program flow of both the module generators and shape functions is devised in a modular and hierarchical manner such that the design of these components for other layout cells can be done easily. The run-time and memory requirements are found to be a minor factor for the system execution. The prototype implementation is also designed with maintainability and code readability in mind.

Chapter 6

Example Layouts

This chapter examines example layouts of I/O pad macros being assembled using the layout templates is described Section 2.1. Individual circuit components are synthesized with the module generators; the pad's layout is assembled manually. Only examples of CMOS compatible buffers are presented. Other layouts for compatible buffers with TTL and ECL signals will be developed in the near future. In addition, an automatic router and floorplanning software for the synthesis problem are under development.

Figures 6.1 and 6.2 show bi-directional CMOS I/O pad circuits laid out for both narrow and wide configurations in the target BiCMOS and CMOS processes. The circuit elements in all layouts have the same specifications. The only distinction among the four layouts is the rectangular aspect ratio of the narrow and wide configurations as well as that of the individual layout blocks.

Some common features include the 10-finger transistors in the output drivers, a p+/n diode with an equivalent active area of $500\text{ }\mu\text{m}^2$ and a $1\text{ k}\Omega$ n-well resistor. Since the input buffers, pre-drivers and control logic circuits can be constructed as standard cells, they are generated by an existing leaf cell generator [12]. Other similarities reflect in the measures implemented for latch-up prevention and ESD protection such as the placement of the snap-back transistor within the NMOS' transistor guard ring. In addition, guard bands are added between the layout of the output drivers and that of the logic block. For layouts generated in the prototype BiCMOS process, the overall dimensions of the narrow and wide configurations are $X = 163\text{ }\mu\text{m}$ and $Y = 586\text{ }\mu\text{m}$, and $X = 340\text{ }\mu\text{m}$ and $Y = 357\text{ }\mu\text{m}$ respectively. The width and height of the narrow and wide pads in the prototype CMOS process are $X = 251\text{ }\mu\text{m}$, $Y = 536\text{ }\mu\text{m}$, and $X = 398\text{ }\mu\text{m}$, $Y = 398.8\text{ }\mu\text{m}$.

With the help of the module generators, the layout design effort can be reduced by 75% compared to a completely manual design. Considering that layout design represents approximately 40% of the overall buffer design cycle, an estimate of 30% savings in the I/O pad design cycle can result from the use of the generators.

FIGURE 6.1 Synthesized layouts of bi-directional CMOS I/O buffers constructed in a $0.8 \mu m$ BiCMOS process.

FIGURE 6.2 Synthesized layouts of bi-directional I/O buffers constructed in a $1.2 \mu\text{m}$ CMOS process.

Chapter 7

Conclusions and Suggestions for Future Research

7.1 Summary

This thesis proposed a design methodology of the layout synthesis system for standard I/O pad macros. The emphasis was on the development and implementation of the shape functions and module generators for the individual building blocks.

Chapter 2 described the relevant issues associated with the design of standard I/O pad macros. These issues include immunity against latch-up, ESD protection, multiple pad configurations and mixed signal levels. Chapter 3 presented the proposed design methodology along with a description of the issues related to process information, floorplanning, routing and compaction. In addition, this chapter proposed a general specification for a floorplan algorithm devised for the layout synthesis of I/O pad macros. Chapter 4 focussed on the description of the module generators and shape functions for the individual special devices. Also, this chapter provided the input parameters that the user can control on selected layout features and described the design constraints imposed on the shape function of each building block. To demonstrate the portability of the module generators, example layouts are presented. Chapter 5 presented the prototype implementation for the module generators which was devised on the basis of three software engineering concepts: abstraction, modularity and hierarchical structure. Finally, Chapter 6 presented example layouts for bi-directional CMOS compatible pads to demonstrate the functionality of the module generators and shape functions. The overall layout was assembled manually but the individual building blocks are synthesized with the module generators and a leaf cell generator.

7.2 Contributions

The contributions of this thesis can be summarized as follows:

- The problems involved in the design of standard I/O buffers are presented along with the design requirements associated with them.
- A design methodology is proposed as a framework for the layout synthesis of I/O pad macros. Two key components in the methodology are the module generators and shape functions.
- Special requirements are provided as a set of specifications for a floorplan algorithm to be designed in the context of placement for I/O pad macros.
- The details of each module generator and shape function associated with the individual circuit elements are described.
- A prototype implementation for the module generators and shape functions is presented along with example layouts synthesized for a BiCMOS and a CMOS process.
- Example layouts of complete CMOS bi-directional pad macros are implemented with the help of the module generators.

7.3 Suggestions for Future Research

There are many opportunities to continue the work described in this thesis. Clearly, a floorplanning, router and compactor could be incorporated with the shape functions and module generators to completely automate the synthesis process. In addition, the resistors can be laid out in an interdigitized configuration for circuits which require closely matched resistors. Furthermore, the layouts for TTL and ECL compatible pads can be created easily with the generators developed in this thesis. More specifically, various buffer topologies can be implemented and compared. A long term vision associated to this thesis is the develop of a synthesis system for I/O pad macros such that the resulting pads have optimum performance measured in terms of layout area, speed, power dissipation as well as process variations.

Glossary of Terms and Symbols

CIF	Caltech Intermediate Format -- A layout description language.
CMOS	Complementary Metal Oxide Semiconductor.
Compaction	Task of compressing the layout in all directions such that the total area is reduced.
Detailed routing	Routing phase which completes point-to-point connections between pins on the layout blocks.
ECI	Emitter Coupled Logic.
EDIF	Electronic Data Interchange Format -- A layout description language.
ESD	Electrostatic Discharge -- Transfer of electrostatic charge between bodies at different electrostatic potentials caused by direct contact or induced by an induced field.
Finger transistor	Individual transistor that composes a larger one. Usually, the finger transistors are connected in parallel to form the large one.
Floorplanning	Placement of layout cells with flexible layout dimensions such that the overall layout area and wiring cost are minimized.
Global routing	Routing phase whereby the connections between layout blocks of the circuit are completed disregarding the exact geometric details of each wire and pin.
Gold doping	Electron doping with gold atoms. Processing technique used to shorten the life of minority carriers to improve immunity to latch-up -- the electron's energy level of gold are closer to the conduction band than that of boron.
Ground bounce	Phenomenon whereby the potential of a supply rail fluctuates due to large current variation. This usually occurs at high operating frequency where a bonding wire's inductance has a voltage drop of Ldi/dt .

Latch-Up	Phenomenon whereby parasitic bipolar transistors are introduced with a presence of a CMOS bulk. The result of this effect is the shorting of the V_{DD} and V_{SS} lines.
LSCR	Lateral Silicon Control Rectifier -- Layout structure used as ESD protection device for I/O pad circuit.
Mid-point threshold	The voltage at which a logic device recognizes a logic high and below which, the device recognizes a logic low.
Module generator	Program which synthesizes the layout geometries of a special layout cell.
MOS	Metal Oxide Semiconductor.
MOSFET	Metal Oxide Semiconductor Field Effect Transistor.
NMOS	N-type MOSFET device.
PMOS	P-type MOSFET device.
Retrograded well	Doping technique where wells have a higher doping concentration at the bottom to reduce the depth of the well. This processing technique can be used to reduce parasitic resistance of a well, R_w .
Salicided region	Self-aligned silicide. Implantation of this material is usually done at contact regions to reduce resistivity.
SCR	Silicon Control Rectifier.
Slew rate	Rate of change of voltage versus time -- V/sec.
Stand-by power dissipation	Situation whereby both PMOS pull-up and NMOS pull-down in a CMOS inverter are on simultaneously because the CMOS transistor cannot recognize the input logic level; e.g. a TTL gate is driving a CMOS inverter -- TTL's $V_{OH}(\min) = 2.4V$ which is too low for the CMOS' $V_{IH}(\min) = 3.5V$ in order to turn off the PMOS pull-up device.
Shape function	Set of equations which summarize the layout dimensions of feasible solutions of a cell based on a set of features.
Trench isolation	Isolation technique to separate devices of opposite types. This processing technique blocks lateral current path to provide a better immunity against latch-up.
TTL	Transistor-Transistor-Logic device.

$V_{IH}(\text{min})$	Minimum input high voltage -- The minimum allowed input high in a logic system. This value represents the guaranteed input high threshold for the device.
$V_{IL}(\text{max})$	Maximum input low voltage -- The maximum allowed input low in a logic system. This value represents the guaranteed input low threshold for the device.

Appendix A

Coding Convention

A program that produces expected results does not necessary imply that it is a quality program. Other criteria must be considered, among them are readability and maintainability. This appendix presents the coding conventions made in the context of development of the prototype program to facilitate readability and hence support for future development.

A.1 Naming of Variables

To distinguish the different types of variables defined in the prototype implementation, the variables name, represented by <VarName>, is preceded by one or two characters. There is, however, an exception where a the name of a global variable is bounded by two asterisks "**".

1. Definition of a flavor	t<VarName>	ex: tResistor, tGuardRing
2. Instance of a flavor	i<VarName>	ex: iResistor, iGuardRing
3. Method of a flavor	m<VarName>	ex: mConstruct_GuardRing_ProcessA
4. Variable method	mv<VarName>	ex: mvConstruct_GuardRing
5. Function	f<VarName>	ex: fRotate_Object_ProcessA
6. Function variable	fv<VarName>	ex: fvRotate_Object
7. Flag	fl<VarName>	ex: fl1stGateFound
8. List of objects	l<VarName>	ex: lGate, lContact
9. Global variable	*<VarName>*	ex: *lLayersToDraw*
10. Layer types	la<VarName>	ex: laMetal1, laNWell

In addition, variables name of functions and methods must be chosen such that:

- a) it describes an action or transformation on an arbitrary object;
Recommended names are: fOpenFiles, fBind_Tech_Methods, fMultiple_ofTen
- b) it does not describe an inverse/negative logic action. Use a positive logic action verb and return NIL if the action fails.

Not recommended names are: fObject_NOTFound, mCannot_Rotate_Object

- c) the action verb must precede the object noun;

Recommended names are: fMake_CIF_Box, mDraw_Polygon

Not recommended names are: fCIF_Box_Creation, mPolygon_Drawing

- d) a capital letter is used for each new word in the variable;

Not recommended names are: fDraw_a_polygon, mcreate_resistor_instance

- e) and if a preposition is required, it must precede a noun.

Recommended names are: mPlace_Contacts_on_Diode, fSnap_to_Grid

A.2 Module Generators and Shape Functions

It is also important to distinguish the meaning of certain terms used in the system to reflect the correct meaning based on the context. The variables WIDTH and LENGTH are the most prone for misuse and could lead to incorrect output.

- a) If these variables are referred to an PMOS or an NMOS:

WIDTH is the width of the transistor.

LENGTH is the length of the gate.

- b) If these variables are referred to the dimensions of a rectangular object:

WIDTH is the vertical dimension of the object.

LENGTH is the horizontal dimension of the object.

A.3 General Conventions

Finally, it is advised to standardize the programming style across the prototype implementation such that any exception or warning with respect to a given function or method is not submerged amid other information. Furthermore, maintainability can be facilitated if the history of the changes made to the software module are recorded.

1. The sequence of information adopted for commenting the functions and methods is as follows:

```
;; -----
;; <State the purpose of the function or method>
;; <Input variables>
;; [<Output variables/Returned values>]
;; [<Warnings/Exceptions/Limitations>]
;; -----
```

2. Maintainability is best achieved if the history of changes made to a software module are recorded. For every modification made to a module, a version and an update number are created along with the reason why the change is required and what has been modified. In addition, the dates the update is opened and closed. Any software modules that has a same update number are backed up under an directory with the update number as search name. At all time, at least three updates are kept in the back up directory. Updates older than six weeks are deleted, except it is a deliverable update. The recommended name for an update number is comprised of three-characters initial of the designer's name followed by an incremental number. For example, valid update numbers are KMN1, KMN2, etc... The update information are written to the beginning of every software module. An example is presented below.

```
;; -----
;; MOSFET.lsp: Module generator for MOS transistors.
;;
;; V1.7    Update: KMN.25    Open: Oct 10, 1992    Close: Oct. 23, 1992
;; - This update is required to fix the translation functions. Modifications were made
;; to fTranslate_Box and fDetermine_Translation_Coordinates.
;; -----
```

3. The distinction between the language syntax and variables is achieved by ensuring all keywords are in upper cases.

Appendix B

Example Source Codes

This appendix illustrates a specific example of source codes written for the module generators. The codes follow the coding convention described in Appendix A as well as the general program flow of a module generator being presented in Section 5.6. The selected example is a generator for a ring. It represents an object with four rectangular stripes connected at 90 degrees of each others.

```

-----
:: Ring.lsp : Ring generator -- This object represents elements having
::           four rectangular stripes connected at 90 degrees of each
::           others.
::
:: By: Kim-Minh Nguyen
::
:: V1.1 Update: KMN.31 Open: June 9, 1993 Close: June 10, 1993
::       - Modify Parsing function to check for NIL condition before
::         checking other conditions.
:: V2.0 Update: KMN.33 Open: June 16, 1993 Close: June 18, 1993
::       - Create target and aspect for mDraw_Stripe_CIFSymbol.
-----

::
:: This method draws a stripe polygon with the specified layer.
::
:: If LayerToDraw is NIL then set the CIF layer for the stripe polygon
:: and draw its geometry to the CIF file. Otherwise, only draw the stripe
:: if its layer is the same as that of LayerToDraw. In addition, do not
:: invoke fSet_CIF_Layer for the latter case because the layer has been
:: set in function fDefine_Device_CIFSymbol.
::
:: INPUT:
:: [LayerToDraw] If NOT NIL then only draw the stripe if its layer is
::               the same as LayerToDraw.
::
-----
(DEFMETHOD (tStripe :mDraw_Stripe_CIFSymbol)
  ;-----
  (&OPTIONAL (LayerToDraw NIL))
  (LET* (
    ;; The two commands below are required to obtain the CIF
    ;; layer of the stripe polygon. These instructions are necessary
    ;; to maintain a technology independent system.
    (StripeLayer (SEND SELF :Stripe StripeLayer))
    (CIFLayer    (fGet_CIFLayer StripeLayer))
    (StripeBox   NIL)
  )
    (IF (OR (EQ LayerToDraw NIL) (EQ LayerToDraw StripeLayer))
      (PROGN ;Then
        ;; Set the layer (writing to CIF file).
        (IF (EQ LayerToDraw NIL) (fSet_CIF_Layer CIFLayer))

```

```

;; Make a list consisting of (Length Width ICenter IDirection) of
;; the stripe polygon. The reason we need to flatten the list is
;; because ICenter and IDirection are themselves a list.
(SETQ StripeBox
  (fFlat_List (LIST (SEND SELF :Length) (SEND SELF :Width)
                    (SEND SELF :ICenter) (SEND SELF :IDirection)
  )))list/flat list/setq

;; Draw a box with dimension and direction taken from the flavor
;; instance.
(fMake_CIF_Box StripeBox)
);progn
);if
);let
);defmethod

;; -----
;; This method computes the geometry for a stripe of a ring. The dimension
;; of this stripe depends on the position it situates with respect to the
;; ring. If the stripe is on the left or right of the ring, then its length
;; is equal to the stripe's width; its width is equal to the sum of the
;; ring's inner width and twice the size of the stripe width. If the stripe
;; is on the top or bottom of the ring, its length is the sum of the
;; ring's inner length and twice the size of the stripe width; its width
;; is equal to the stripe's width.
;; The center position is computed with respect to the center of the ring.
;; When the dimension is computed, the stripe instance is added to
;; :lElement of the ring instance.
;;
;; INPUT:
;; StripePos      The position of the stripe with respect to the center of
;;                the ring.
;; -----
(DEFMETHOD (tRing :mCompute_RingStripe_Geometry) (StripePos)
  ;-----
  (LET* ( ;; The specifications of the ring.
    (RingLayer      (SEND SELF :RingLayer))
    (InnerLength    (SEND SELF :InnerLength))
    (InnerWidth     (SEND SELF :InnerWidth))
    (StripeWidth    (SEND SELF :StripeWidth))
    (InnerLengthOver2 (/ InnerLength 2))
    (InnerWidthOver2 (/ InnerWidth 2))
    (StripeWidthOver2 (/ StripeWidth 2))
    (IDirection     (SEND SELF :IDirection))
    (IRingCenter    (SEND SELF :ICenter))
    (RingCenterX    (FIRST IRingCenter))
    (RingCenterY    (SECOND IRingCenter))

    ;; Local variables for a stripe instance.
    (iStripe        NIL)
    (Stripe_L       0) (Stripe_W 0)
    (StripeCenterX  0) (StripeCenterY 0)
  )

  (CASE StripePos
    ((LPt RPt) ;; The stripe is that on the left or on the right of the
      ;; ring. The stripe's length and width for both cases
      ;; are the same.
      (SETQ Stripe_L StripeWidth)
      (SETQ Stripe_W (+ InnerWidth (* 2 StripeWidthOver2)))

      ;; Compute the stripe center's X coordinate.
      (IF (EQ StripePos 'LPt)
        (SETQ StripeCenterX (- RingCenterX InnerLengthOver2 StripeWidthOver2))
        (SETQ StripeCenterX (+ RingCenterX InnerLengthOver2 StripeWidthOver2)))

```

```

);if

;; The stripe center's Y coordinate.
(SETQ StripeCenterY RingCenterY)
);lpt or rpt

((TPt BPt) ;; The stripe is that on the top or at the bottom of the
;; ring. The stripe's length and width for both cases
;; are the same.
(SETQ Stripe_L (+ InnerLength (* 2 StripeWidth)))
(SETQ Stripe_W StripeWidth)

;; The stripe center's X coordinate.
(SETQ StripeCenterX RingCenterX)

;; Compute the stripe center's Y coordinate.
(IF (EQ StripePos 'TPt)
  (SETQ StripeCenterY (+ RingCenterY InnerWidthOver2 StripeWidthOver2))
  (SETQ StripeCenterY (- RingCenterY InnerWidthOver2 StripeWidthOver2)))
);if
);tpb or tpb

(OTHERWISE
  (RETURN-FROM :mCompute_RingStripe_Geometry
    (fSwError 'mCompute_RingStripe_Geometry '(Invalid StripePos) StripePos))
);otherwise
);case

;; Make a stripe instance. The stripe layer is that of the ring.
(SETQ iStripe (MAKE-INSTANCE 'tStripe
  :StripeLayer RingLayer
  :Length Stripe_L
  :Width Stripe_W
  :Center (LIST StripeCenterX StripeCenterY)
  :IDirection IDirection)
);make-instance

);setq

;; Add this stripe to the ring's list of elements.
(fPush_Instance_to_ICollection iStripe SELF)
);let
);defmethod

;; -----
;; This method computes the geometry of the four stripes of the ring and
;; updates the outer dimension of the ring. The stripes' dimension is
;; determined based on its position with respect to the center of the
;; ring.
;; The ring is computed with reference position being its center because
;; the ring should be concentric around the object it encircles.
;;
;; INPUT:
;; [NoRingStripe_Pos] If this parameter is NOT NIL then there is no ring
;; stripe on the position indicated by this parameter.
;; This parameter is required when a p+ diode is
;; constructed. Refer to flavor tDiode.
;; -----
;;
(DEFMETHOD (tRing :mCompute_Ring_Geometry)
  (&OPTIONAL (NoRingStripe_Pos NIL))
  (LET (
    ;; The specifications of the ring.
    (InnerLength (SEND SELF :InnerLength))
    (InnerWidth (SEND SELF :InnerWidth))
    (StripeWidth (SEND SELF :StripeWidth))

```

```

;; A ring is composed of four stripes with same layer as the ring
;; layer. These stripes are built counter-clockwise starting from
;; the left.
(StripePos (LIST 'LPt 'TPt 'RPt 'BPt))

;; The position of the stripe to be built.
(StripePos NIL)
)
;; Compute the dimension of the four stripes.
(DOTIMES (Index 4)
  ;; Get the stripe position.
  (SETQ StripePos (NTH Index IStripePos))

  ;; Only create a stripe for the ring if its position is different
  ;; than NoRingStripe_Pos. The latter parameter is required when
  ;; a p+ diode is constructed. Refer to Diode.lsp.
  (IF (NOT (EQ StripePos NoRingStripe_Pos))
    ;; The following method creates a local tStripe instance with
    ;; the same layer as that of the ring type. The geometry of
    ;; the stripe is computed based on the specifications of the
    ;; ring. The stripe instance is then added to the ring's
    ;; :Element.
    (SEND SELF :mCompute_RingStripe_Geometry StripePos)
  )if
);dotimes

;; Update the outer length and outer width of the ring.
(SEND SELF :SET-Length (+ InnerLength (* 2 StripeWidth)))
(SEND SELF :SET-Width (+ InnerWidth (* 2 StripeWidth)))

);let
);defmethod

-----
;; This method parses the flavor instance to ensure that the required
;; information about the instance is properly initialized.
;;
;; RETURNS: If there is no problem, T is returned. Otherwise, a software
;; error is generated and NIL is returned.
;;
-----
(DEFMETHOD (tRing :mParse_Ring_Instance) ()
  ;-----
  (LET ((RingLayer (SEND SELF :RingLayer))
        (InnerLength (SEND SELF :InnerLength))
        (InnerWidth (SEND SELF :InnerWidth))
        (StripeWidth (SEND SELF :StripeWidth)))
    (COND ((EQ RingLayer NIL)
      (fSwError 'mParse_Ring_Instance '(Invalid RingLayer) RingLayer))
      ((OR (EQ InnerLength NIL) (ZEROP InnerLength))
        (fSwError 'mParse_Ring_Instance
          '(Invalid InnerLength) InnerLength))
      ((OR (EQ InnerWidth NIL) (ZEROP InnerWidth))
        (fSwError 'mParse_Ring_Instance
          '(Invalid InnerWidth) InnerWidth))
      ((OR (EQ StripeWidth NIL) (ZEROP StripeWidth))
        (fSwError 'mParse_Ring_Instance
          '(Invalid StripeWidth) StripeWidth))
      (T (EVAL T)))
  );cond
);let
);defmethod

```



```

-----
;; This function builds a ring based on the specifications below.
;;
;; INPUT:
;; RingLayer           The layer of the ring.
;; InnerLength         The inner length of the ring.
;; InnerWidth          The inner width of the ring.
;; StripeWidth         The width of the four stripes of the ring.
;; IRingCenter         XY coordinates of the ring's center.
;; IDirection          The direction vector of the ring.
;; [NoRingStripe_Pos] If this parameter is NOT NIL then there is no ring
;;                    stripe on the position indicated by this parameter.
;;                    This parameter is required when a p+ diode is
;;                    constructed. Refer to flavor tDiode.
;;
;; RETURNS: The ring instance is returned.
-----
(defun fBuild_Ring
  (RingLayer InnerLength InnerWidth StripeWidth IRingCenter
   &OPTIONAL (IDirection '(1 0)) (NoRingStripe_Pos NIL))

  (let ((iRing NIL))
    ;; Make an instance with the required entries initialized with the input
    ;; parameters.
    (setq iRing (MAKE-INSTANCE 'tRing
      :RingLayer RingLayer
      :InnerLength InnerLength
      :InnerWidth InnerWidth
      :StripeWidth StripeWidth
      :ICenter IRingCenter
      :IDirection IDirection
    );make-instance
    );setq

    ;; Make sure the information required to build this ring are properly
    ;; initialized.
    (SEND iRing :mParse_Ring_Instance)

    ;; Compute the ring geometry with the reference point being its center.
    (SEND iRing :mCompute_Ring_Geometry NoRingStripe_Pos)

    ;; Return the ring instance.
    iRing
  );let
);defun

```

Appendix C

Design Processes

The layout for the I/O pad macros is constrained by the process technology which is available for production of the wafers. Two prototype processes have been used to demonstrate the portability of the prototype system. The first design process is a BiCMOS technology developed by Northern Telecom Electronics Inc. [8]. It is a 0.8 μ m twin-well BiCMOS process which features buried layers with three levels of metal interconnect and three layers of polysilicon. Buried layers are used to minimize parasitic resistance, hence reducing lateral current in the substrate. The first and second metal layers are made of tungsten whereas the top layer is composed of an aluminum-silicon-copper alloy. Since tungsten plugs are employed, and a planarization process is used in this process, the interconnect contacts, vias and topvias can be stacked. The first polysilicon layer is used to define MOSFET gates and capacitor bottom plates, the second layer is utilized to define poly-poly capacitors, and the third layer is used to define NPN emitters and linear resistors.

The second process is a CMOS technology and is also developed by Northern Telecom Electronics Inc. [5]. It is a 1.2 μ m CMOS technology which features twin-well, double-metal and double-polysilicon layers. One of the polysilicon layers forms MOSFET transistor gates, interconnect and capacitor bottom plates while the second polysilicon layer is used to form the top plates of capacitors. The two metal layers are made of aluminium. Stacked contacts or vias are not allowable in this target process.

References

- [1] Alvarez A., *BiCMOS Technology and Applications*, Kluwer Academic Publishers, 1989.
- [2] Avery L., "ESD Protection Structure Issues and Design for Custom Integrated Circuits", *Proceedings of the I.E.E.E. Custom Integrated Circuits Conference 1988*, pp. 27.1.1 - 27.1.4.
- [3] Brooks R., *Programming in Common LISP*, John Wiley & Sons, 1985.
- [4] Buchanan J., *BiCMOS/CMOS Systems Design*, McGraw-Hill, 1991.
- [5] Brown D, Scott A., *Design Rules and Process Parameters for the Northern Telecom CMOS4S Process*, Canadian Microelectronics Corporation, Report IC90-01, February 1990.
- [6] Braiman M., Kuppinger J., "Automated Generation of Custom Pad Cells for ASICs", *Proceedings of the I.E.E.E. Custom Integrated Circuits Conference 1991*, pp. 22.6.1-22.6.4.
- [7] Embabi S., Bellaouar A., Elmasry M., *Digital BiCMOS Integrated Circuit Design*, Kluwer Academic Publishers, 1993.
- [8] Hadaway, R. et al., "A Sub-Micron BiCMOS Process for Telecommunications Circuits", *Proceedings of 21st European Solid-State Device Research Conference, ESSDERC 91*, Lausanne, September 1991.
- [9] Hara H et al , "A 350 ps 50K 0.8um BiCMOS Gate Array with Shared Bipolar Cell Structure", *Proceedings of the I.E.E.E. Custom Integrated Circuits Conference 1991*, pp. 8.5.1-8.5.4.
- [10] Kuroda T. et al., "Automated Bias Control (ABC) Circuit for High-Performance VLSI's", *I.E.E.E. Journal of Solid-State Circuits*, Vol. SC-27, No. 4, April 1992, pp. 641-648.
- [11] Lefebvre M.C., *Micronet/CMC BiCMOS Design Environment*, Chapter 10, 1993.
- [12] Lefebvre M.C., Skoll D., "PicassoII: A CMOS Leaf Cell Synthesis System", *Proceedings of the Microelectronics Center North Carolina (MCNC) International Workshop on Layout Synthesis*, May 1992.
- [13] Lengauer T., *Combinational Algorithms for Integrated Circuit Layout*, John Wiley & Sons, 1990.
- [14] McAteer O.J., *Electrostatic Discharge Control*, McGraw-Hill, 1989.
- [15] Nguyen K.M., Lefebvre M.C., "A Family of Module Generators for the Layout Synthesis of I/O Buffers", *Proceedings of the VLSI93 Conference*, 1993, pp. 1.3.1-1.3.10.

- [16] Odaka M. et al., "A 12ns / 350 mW 16Kb ECL Compatible RAM", *E.S.S.C.I.R.C. 1895, 11th European Solid State Circuits Conference*, 1985, pp. 166-176.
- [17] Pickles N.S., *Design, Modeling, and Optimization of ECL Interface Circuits for BiCMOS Integrated Systems*, M.Eng. Thesis, Carleton University, June 1993.
- [18] Pickles N.S., Lefebvre M.C., "ECL I/O Buffers for BiCMOS Integrated Systems: A Tutorial Overview", *I.E.E.E. International ASIC Conference and Exhibit*, 1993, To be published.
- [19] Pressman R., *Software Engineering - A Practitioner's Approach*, McGraw Hill, 1992.
- [20] Sedra A., Smith K., *Microelectronics Circuits*, Saunders College Publishing, 1991.
- [21] Sherwani N., *Algorithm for VLSI Physical Design Automation*, Kluwer Academic Publishers, 1993.
- [22] Yang E., *Fundamentals of Semiconductor Devices*, McGraw-Hill, 1978.
- [23] Weste N., Eshraghian K., *Principles of CMOS VLSI Design - A systems Perspective*, Addison-Wesley, 1988.
- [24] Winston P., Horn B., *LISP*, Second Edition, Addison-Wesley Publishing Company, 1984.
- [25] Wong A. et al., "A High Density BiCMOS Direct Drive Array", *Proceedings of the I.E.E.E. Custom Integrated Circuits Conference 1988*, pp. 20.6.1-20.6.3.
- [26] Woodruff R. et al., "A New Latch-Up Resistant CMOS Buffer with a Novel TTL Input Stage", *Proceedings of the I.E.E.E. Custom Integrated Circuits Conference 1984*, pp. 584-587.
- [27] Xia H., Lefebvre M.C., "Optimized-Based Placement Algorithms for BiCMOS Leaf Cell Generation", *I.E.E.E. Bipolar/BiCMOS Circuit and Technology Meeting*, To be published.

END

2 0 0 7 - 9 4

FIN