

University of Science and Technology at Zewail City
Department of Nanotechnology and Nano-electronics Engineering

Design Aiding and Layout Automatic Synthesis for Analog Circuit

by

Ali Mohamed Fathalla

Moaz Alaa Elshamy

Mohamed Hamdy Nofal

Supervised by: Dr. Hassan Mostafa Hassan

In collaboration with: Si-vision IC solutions corporation as part of Made In Egypt initiative

Senior Design Project Thesis submitted to fulfill the Bachelor's Degree

June 2019



Abstract

Analog circuits are essential part of nearly all of the IC industry making nearly 15% of the share of its market. The process of analog IC design is known to be time consuming and error-prone. According to a statistical data published by EDA weekly on 2005, Analog design takes nearly 40% of the design effort and causes about 50% of the errors found. This is in spite of the fact that analog components represents usually only 3% of the area in system-on-chip designs.

With that said, unfortunately, the EDA solutions made to facilitate the work of analog designers are very limited and outdated. This was due to the focus given through the last decades for digital circuits in accordance with Moore's law. Also, the fact that analog design is very sensitive to various effects and involves multiple designing constraints that are not found in its digital counterpart. Analog design is of a heuristic nature and knowledge intensive. Luckily, with more reports on how analog design has become the bottle neck of IC design, now, EDA companies are shifting their focus and intensifying their efforts to automate Analog circuit design.

Analog design consists primarily two paths, top-bottom electrical path and bottom-up physical path. The physical path consists of layout generation and verification. Layout generation has reported to cause nearly 30% deviation of the electrical results in small nodes (20 nm and below). Hence, it is of very critical importance to seek automation solutions for the layout generation problem.

This thesis summarize the work done over a period of nearly 6 months in collaboration with a promising IC solution corporation, Si-Vision. This was out of the need to develop inexpensive automation solutions that can benefit the company on the short term. The book is organized as follow:

- Chapter 1: Introduction to Analog Circuit Layout Generation
- Chapter 2: Literature Review on the-State-of-the-Art Solutions Found in Literature
- Chapter 3: Proposed Automation Flow
- Chapter 4: Results & Discussions
- Chapter 5: Conclusion & Future Work

Acknowledgements

First of all, we would like to acknowledge the continuous help and support provided by our great mentor and supervisor Dr. Hassan Mostafa. His steady and continuous guiding throughout the whole project, in addition to his comments and discussions were of invaluable help.

We would like to express our gratitude for the team of brilliant team of engineer at Si-Vision IC Solution Corporation for their help, support and for the valuable explanations and discussions we received throughout the project including but limited to Eng. Mostafa Nashaat, Eng. Islam Nashaat, Eng. Shrouk Shaafy and Eng. Moustafa Hussien. We would like to thank, in particular, Eng. Fady Atef for his eagerness and enthusiasm towards the work. He has proven to have a great command of leadership.

We would like also to thank the Made-In-Egypt (MIE) initiative for coordinating with Si-Vision to come up with such a unique program that really boosts the collaboration between academia and industry.

Last but not least, we would like to thank our beloved families and friends for their continuous support and sincere love that gave us the power to do our best.

Contents

Abstract.....	I
Acknowledgements.....	II
Contents.....	III
List of Figures.....	VI
List of Tables.....	X
Abbreviations.....	XI
List of Symbols.....	XIII
Gantt Chart.....	XIV
Chapter 1: Introduction.....	1
1.1 Analog and Mixed Signals (AMS) ICs.....	1
1.2 Analog Circuit Layout Generation and Verification.....	4
1.3 Motivation for Analog Design Automation.....	8
1.4 Analog Layout Automation.....	9
1.5 Advances to The-State-of-Art.....	11
1.6 Conclusion.....	12
References.....	13
Chapter 2: Literature Review on the State-of-the-Art Analog Layout Automation Solutions.....	15
2.1 Placement.....	16
2.1.1 Analog Topological Constraints.....	16
2.1.2 Floorplan Representations.....	17
2.1.2.1 Absolute Representation.....	18
2.1.2.2 Topological Representations.....	19
2.1.2.2.1 Slicing Representations.....	20
2.1.2.2.1 Non-Slicing Representations.....	20

2.1.3 Modern analog placement challenges and schemes.....	23
2.1.3.1 Pareto Front of Placement Solutions.....	23
2.1.3.2 Thermal-Driven Placement.....	24
2.1.3.3 Current-Driven Placement.....	24
2.1.4 Stochastic Optimizer.....	25
2.1.5 Commercial Solutions.....	26
2.2 Routing.....	26
2.2.1 From Netlist to Pathfinder.....	27
2.2.2 Electromigration and IR-drop.....	27
2.2.3 Electromigration-Aware Approaches.....	28
2.2.4 Wiring Symmetry.....	29
2.3 Complete Layout Generation tools.....	29
References.....	33
Chapter 3: Proposed Automation Flow.....	40
3.1 Building Blocks Recognition Algorithm.....	41
3.2 Pattern Generator & Physical Aspect Ratio Estimation.....	44
3.2.1 Common Centroid Matching.....	45
3.2.2 Interdigitization Matching.....	46
3.3 B*trees generation & enumeration.....	47
3.3.1 Constraints in analog layout.....	47
3.3.2 Different representation.....	49
3.3.3 Advantages.....	50
3.4 Linear-Programming-Guiding Constraints Generation.....	52

3.5 Linear Programming.....	53
3.6 Physical Placement by Means of Parametric Cells Evaluation.....	56
3.6.1 Current Mirror.....	56
3.6.2 Differential Pair.....	60
3.7 Conclusion.....	61
References.....	62
Chapter 4: Results & Discussion	63
4.1 Recognizing and constraining basic blocks.....	64
4.2 Basic blocks pattern generation and dimension estimation.....	65
4.3 Formalization of constraints and solving them.....	66
4.4 Detailed placement and routing.....	67
4.5 Conclusion.....	70
Chapter 5: Conclusion and Future Work.....	71
Appendix.....	73
A1: Concept of Layout.....	73
A2: Classification of floorplan representations for analog design automation.....	81

List of Figures

Figure 1.1: AMS Design flow (adopted from [7]).....	3
Figure 1.2: Layout generation and verification process flow.....	5
Figure 1.3: (Upper) current mirror layout using common centroid technique & (bottom) differential pair layout using interdigitization technique.....	8
Figure 1.4: Sample result from Plantage (adopted from [14]).....	12
Figure 2.1: Common topological constraints. A) Matching pattern B) Symmetry C) Proximity (adopted from [1]).....	17
Figure 2.2: Layout example and its representation is slicing tree (adopted from [1]).....	20
Figure 2.3: Non-slicing topology example known as wheel (adopted from [1]).....	21
Figure 2.4: a) Layout topology example b) Its representation in sequence-pair c) O-tree d) B*tree.....	22
Figure 2.5: Comparison between traditional shape function (above) and enhanced shape function (below).....	24
Figure 2.6: Chronological representation of analog layout generators (adopted from [63]).....	32
Figure 3.1: Purposed analog circuit design and layout placement automation flow.....	40
Figure 3.2: Analog circuits' hierarchical construction.....	41
Figure 3.3: A) Correct Multiple Identification B) Wrong Multiple Identification.....	43

Figure 3.4: Quarter Cell: (3) is the flipped version of (2) & (4) is the flipped version of the shaded part.....	45
Figure 3.5: Cross Quad technique [1].....	45
Figure 3.6: Finger by finger matching case example: two transistors no source sharing	46
Figure 3.7: 2 fingers by 2 fingers matching case example: Two transistors matched by 2 fingers shared.....	47
Figure 3.8: Common Centroid Matching pattern.....	48
Figure 3.9 Symmetric placement.....	49
Figure 3.10 Symmetric placement and routing.....	49
Figure 3.11: B*-tree example.....	50
Figure 3.12: Compact placement.....	50
Figure 3.13: SSFG graph building.....	52
Figure 3.14: A schematic of a circuit with the building blocks recognized.....	53
Figure 3.15: B*-tree to VCG.....	55
Figure 3.16: B*-tree to HCG.....	55
Figure 3.17: Current Mirror Automation Flow Chart.....	56
Figure 3.18: Common centroid by means of quarter blocks.....	57
Figure 3.19: Common centroid by means of quarter blocks in case of odd number of rows.....	57
Figure 3.20: Device instantiation.....	59

Figure 3.21: Electromigration.....	59
Figure 3.22: Interdigitized diff pair.....	60
Figure 4.1 Analog circuit schematic requiring different type of constraints.....	63
Figure 4.2: Result of recognized blocks.....	64
Figure 4.3: Basic blocks identified.....	64
Figure 4.4: Symmetric nets identified.....	65
Figure 4.5: Virtual placement of the basic analog blocks	67
Figure 4.6: Layout of the whole circuit.....	68
Figure 4.7: Current mirror pcell evaluated.....	68
Figure 4.8: The details of the routing.....	69
Figure 4.9: The pcell of the differential pair.....	69
Figure A1: Size effect on error.....	73
Figure A2: Printability Issues in Lithography.....	74
Figure A3: optical proximity correction.....	75
Figure A4: Double patterning technology.....	75
Figure A5: Devices to be matched.....	77
Figure A6: Choosing matching element of 250 ohm.....	78
Figure A7: Choosing matching element of 1 k Ω	79

Figure A8: Common centroid.....78

Figure A9: Interdigitization.....79

Figure A10: Latch Up.....79

List of Tables

Table 2.1: Comparison between state-of-the-art works on layout-aware sizing with special emphasis to layout-related data included in-loop.....	31
Table 2.2: Optimization algorithms used in analog placement automation tools.....	32
Table 3.1: Analog circuits' building blocks library.....	42
Table 3.2: Multiple block recognition solution.....	43
Table A.1: Classification of floorplan representations for analog design automation.....	81

Abbreviations

IC: Integrated Circuit

CAD: Computer-Aided Design

AMS: Analog and Mixed Signals

WSTS: World Semiconductor Trade Statistics

SoC: Systems-On-Chip

SNR: Signal-To-Noise Ratio

ADC: Analog-To-Digital Converter

DRC: Design Rules Checking

LVS: Layout versus Schematic

PEX: Parasitics Extraction

SMT: Satisfiability Modulo Theories

Z3: Microsoft SMT Logic Solver

CDL: CAD file extension used to visualize and store design information for a 3D product such as lines and geometry

B*tree: Modified Binary Tree (Data Structure Type)

VCG: Vertical Constraint Graph

HCG: Horizontal Constraint Graph

SMFL: Simple Fast Media Library (C++ graphics library)

CM: Current Mirror

Diff. Pair: Differential Pair

P-cell: Parametrized Cell

X: Dummy Transistor

PNC: Pin-Net Connection

CP: Cascode pair

DP: Differential pair

LS: Level Shifter

2TCM: 2T-Current Mirror

ML: Mirror Load

WCM: Wilson Current Mirror

VR: Voltage Reference

CC: Cross coupled

4TCM: 4-Transistor Current Mirror

DPNC: Device PNC

TCG: Transitive Closure Graph-based

BSG: Bounded-Sliceline Grid

O-tree: Ordered tree

HB*tree: Hierarchal B*trees

OPC: Optical Proximity Correction

DPT: Double Patterning Technology

WPE: Well Proximity Effect

PVT: Process, Voltage and Temperature

List of Symbols

A_p : Area proportionality constants

S_p : Separation proportionality constant

D_x : Separation distance

$\sigma^2(\Delta p)$: The mismatch induced deviation

T : Temperature

k : Boltzmann constant

J : Current density

X_c : X axis centroid for a device/block

w : Width of a device/module

l : Length of a device

h : Height of a module

m : Module

x_m, y_m : Lower left axis co-ordinates

V_{th} : Threshold voltage

X_{dt} : Maximum space charge width

ϕ_{fp} : Potential difference between interinsic Fermi level and Fermi level

ϕ_{ms} : Metal work function

N_a, N_d : Doping concentration

Q_{ss} : Charge density in the oxide

Q_{sd} : Charge in the space charge region

Gantt Chart

ID	Task	Oct-18		1-Nov		Dec-18		1-Jan		Feb-19		1-Mar		Apr-19		1-May		Jun-19		
		1 st half	2 nd half	1 st half	2 nd half	1 st half	2 nd half	1 st half	2 nd half	1 st half	2 nd half	1 st half	2 nd half	1 st half	2 nd half	1 st half	2 nd half	1 st half	2 nd half	
1	Team building and project goals set	█																		
2	Literature Review on analog layout automation methods		█	█																
3	Learning manual analog layout generation				█	█	█													
4	Learning SKILL scripting language								█											
5	Revise the work of Plantage to build upon it (Top flow building)									█	█									
6	Developing codes for analog building blocks recognition and parametrized cells construction										█	█								
7	Developing codes for constraints generation and B*tree generation													█	█					
8	Developing codes for B*tree enumeration and linear solver programming																█	█		
9	Testing and integrating the results																		█	█
10	Writing the thesis																			█

“Dedicated to the Memory of Nour El-din”

Chapter 1

Introduction

During the last few decades, exponential advancements were observed in the domains of consumer electronics such as home appliances and smart phones, industrial electronics for instance smart grids, industrial automation and motion control, medical applications, meteorological and oceanographic systems, defense and aerospace sectors in addition to automotive industries. Semiconductor industries, integrated circuits (ICs) in particular, were in the heart of this huge growth in nearly all sectors of technology. The ICs market expanded from a relatively small market of \$10 billion in 1980 into a gigantic market of \$393 billion in 2018 [1]. ICs' overall performance have steadily increased mainly due to the exponential increase in transistors' density allied with transistors' cost reduction anticipated by Moore's law in 1975 [2]. This advancement is mainly the fruit of fabrication techniques development.

Currently, designers are equipped with the means to build very large, extremely complicated multimillions (or even billions [3]) circuitry that can implement complete systems on a single chip. Given the strict time-to-market constraints, low power, low cost, high speed and new functionalities requirements, the accomplishment of this task is nearly impossible without computer-aided design (CAD) tools assisting the designer throughout the whole process.

1.1 Analog and Mixed Signals (AMS) ICs

AMS ICs constitutes nearly 15% of the size of ICs' market (~ \$59 billion) according to World Semiconductor Trade Statistics (WSTS) in 2018. Although digital circuits constitutes the vast majority of functions in today's ICs, AMS circuitry is and will always be the link between the external world and digital circuits. This is due to the continuous-value nature of the external world.

In modern ICs used in telecommunications and multimedia application, the integration of both digital and AMS blocks on the same systems-on-chip (SoC) is a common procedure [4, 5].

This continuous-value nature of signals processed by analog circuits makes them more susceptible to noise, and process variation than digital circuits leading to a much complicated, time consuming and error prone designing process. Following is some typical blocks expected to remain analog forever [6]:

- On the input side of a systems, received analog signals must be sensed, amplified and filtered to a satisfactory level that allows for digitalization with low signal-to-noise ratio (SNR) and distortion ratio. Hence, typical front-end interfaces in sensors i.e. microphones and communication receivers are and will remain analog.
- On the output side of a system, the digitally processed signal must be reconverted to an analog signal strong enough to drive the output load with low distortion. This can be found in typical communication transmitters and loudspeakers.
- Mixed signals circuits such as analog-to-digital converters (ADCs), sample-and-hold and frequency synthesizers. They serve usually as an input/output interface between the system and its digital processing parts.
- Crystal oscillators, voltage and current references circuitry.
- Very-high performance state-of-the-art digital circuits follows a similar customization and design flow to AMS circuits.

The well-accepted common flow of AMS circuit design can be illustrated in figure 1.1. It consists of two main flows: top-down iteration to meet the design specifications and bottom-up flow to generate the layout and verify the design.

It is possible to adopt a hierarchal methodology to first explore the system architecture and understand how to optimize it using mainly behavioral simulations without the need to start detailed circuit/device level implementation. This increases the chance of design success and reduce the time required for the whole design process. Yet, the parasitics added by the layout and process variations, especially with smaller fabrication technology nodes, forces the designers to reiterate many times in order to reach the required specifications.

To fully understand the designing flow, the main two designing flow is explained in a greater details:

- Top-down flow is mainly the electrical optimization path including circuit topology selection, circuit sizing and verification.
- Bottom-up flow is mainly the physical synthesis path including layout generation and detailed verification (after layout parasitics extraction).

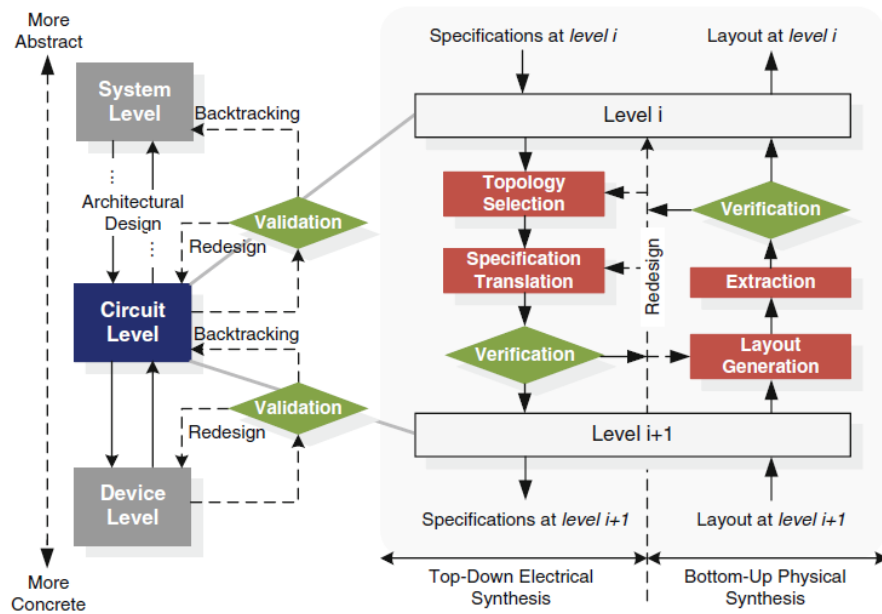


Figure 1.1: AMS Design flow (adopted from [7])

Topology selection is the step where the most appropriate system or circuitry is chosen in order to meet both the functionality and specification of the system required. The topology could be either chosen from existing famous topologies or synthesized. Specification translation is the process where the selected topology is customized to meet the specifications. This step is reduced to circuit sizing at the lowest level of abstraction. First, the specifications translation is verified by behavioral simulation before going down in the hierarchy to circuit sizing. After the sizing, the specifications are verified by means of electrical simulations. Multiple iterations are usually required until satisfactory results are met.

Through years of refinement and development, mature CAD tools are fundamentally used to aid the designer to successfully size the circuit. They are also used to verify the design outputs

by means of both behavioral simulations and electrical simulations. Following are some of the mature tools in use today: Mentor Graphics' ADiT, Questa and Eldo, Synopsys' HSPICE, nanoism and HSim, Cadence's Spectre, ngspice and SMASH. Many of those tools are starting to adopt automation solution such as Cadence's virtuoso Custom Design Platform GXL which has a circuit optimizer feature, Soldo's Fast process, voltage and temperature (PVT), Fast Monte Carlo and High-Sigma Monte Carlo, MunEDA's WCkeD™ and Synopsys frameworks after the acquisition of both Analog Design Automation Inc. Genius product line and Magma Titan in 2004 and 2012 respectively. Yet, most of those solutions are only applicable at cell level only.

In the bottom-up path, the layout is generated in order to be used to manufacture the masks used in fabrication. The layout generation is the process of the drawing the physical geometrical shapes of the circuit. It must obey strict rules from the manufacturer related to yield optimization and prevention of process adversaries such as under/over cut, features fusing and induced wafer fractures. Those rules are generally more restrictive in smaller technology nodes. After the design is laid out, parasitics added from the physical design are extracted, also, the design must be tested for each wafer corner to account for variations. All those parameters are taken into account and verification steps i.e. post-layout simulations are made to assure that the design adheres to the required specifications. In real world, backtracking and redesign are often required to compensate for the added parasitics' effects.

Some of the mature layout editing CADs includes Mentor Graphics' IC station Layout, Synopsys' Galaxy Custom Designer LE and Cadence Virtuoso Layout Editor. Also, Design Rules Checking (DRC), Layout Versus Schematic (LVS) verification and Parasitics Extraction (PEX) can be done by means of different CADs such as Mentor Graphics' CALIBRE, Synopsys' Hercules and Cadence's DIVA and Assura.

1.2 Analog Circuits Layout Generation and Verification

Layout generation and verification process flow is shown at figure 1.2. It consists of seven main steps, schematic and constraints, floor planning, routing, physical verification, PEX, post-layout simulation and finally, signing off.

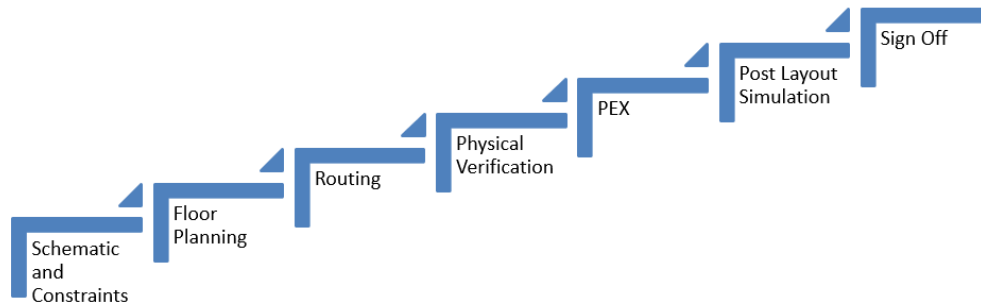


Figure 1.2: Layout generation and verification process flow

First, the designer starts by checking the schematic and whether it meets the physical constraints for example if a differential pair circuit are balanced. Upon the checking of the schematic, the floor planning stages starts. Floor planning aims at organizing the chip in accordance with its functionality, constraints and specifications. It enables the building of neat and efficient chip with targeted interfaces, minimum area, smooth balanced routing and minimum parasitics. The floor planning consists of main four steps:

- Area estimation where an estimate of the area is calculated and the width, length, number of fingers and multipliers (known as transistor folding) of each transistors are determined.
- Pins/Devices placement which aims at gathering common/communication groups together, separate noisy and quiet blocks, place the blocks in a manner that serves the targeted interfaces, keep blocks with the same power domain together, symmetric placement for blocks that requires symmetric routing...etc.
- Power/signal planning where the main power and channel routes are defined. Care must be given to the top level integration, current capabilities of the wires which in turn reflects on their width and path, critical routes and shielding and routing channels capacity.
- Final placement where the placement is finalized and routing scenarios are rechecked.

After the floor planning step is finalized, the blocks are routed. It is important in the routing to know how it would affect the performance, which metal layer would be chosen for each route and

the relationship between the floor plan and routing i.e. symmetric routing. Following is some of the important concepts in routing:

- Parasitic capacitance: parasitic capacitance can be calculated semi-analytically using different models [8][9]. It is worthy to mention that parasitic capacitance may severely limit the performance especially at high frequencies. Common tips are: reduce the wire length in order to reduce its overlap with other wires and/or the substrate, use higher metals for critical signals as they suffer less from parasitic capacitance, use minimum metal width and maximum oxide thickness when possible and avoid wires overlapping.
- Parasitic resistance: parasitic resistance is determined using the sheet resistance of each metal layer supplied by the manufacturer. Parasitic resistance must be compatible with the required voltage drops. To reduce parasitic resistance, it is common to use wider metals, metal stacks and higher metal levels.
- Supply noise: in order to reduce supply noise, separate supplies/grounds are used for each domain. Also, decoupling capacitors are used between the power and ground rails.
- Coupling noise: shielding is used to reduce coupling noise. Shielding could be either lateral or all around depending on the available area, metal levels and signal importance.
- Electro-migration: electro-migration is the phenomena of molecular displacement of atoms caused by the flow of electrons over time leading to wires cut eventually. It increases with current density, temperature and smaller technologies. Hence, the current capabilities of each wire must match with its width in order to avoid electro-migration.
- Antenna effect: antenna effect is a phenomena that occurs when a metal wire is connected to transistor gate causing plasma-etch like effect during fabrication that could lead to oxide breakdown. In order to solve antenna effect, high level metal jumpers and antenna diodes are used.

After the design is placed and routed, physical verification is done. The most two important verifications are DRC and LVS. DRC is technology-dependent and consists of geometrical rules

imposed by the manufacturer due to the process limitations in order to maximize the yield. LVS, on the other hand, checks whether the schematic matches the layout in terms of connections, number of devices and their type. Finally, parasitics are extracted and post-layout simulations are made to compare the physical real performance of the circuit. Iterations of redesign and backtracking may be needed to compensate for the parasitic effects.

It is also worthy to introduce some of the techniques used in layout placement and routing in order to comply with matching constraints. Mismatches mainly arises from two main causes:

- Process variations such as mask production and alignment, lateral diffusion, over/under etching, boundary conditions and non-uniformities in the process i.e. non-uniform doping. This in terms affects the width-to-length ratio of the transistors and the threshold voltage.
- Gradients such as current flow orientation (recommended to be fixed throughout the whole chip), stresses and thermal gradients.

In that context, matching is the process of making certain circuit blocks approximately exposed to the same conditions so that their desired functionality and/or specifications are not altered. Matching patterns could be generated using software guided by the mathematical descriptions of the effects such as stress and temperature using optimization algorithms such as simulated annealing and genetic algorithms [10][11]. This is usually important in large and sensitive designs. However, in many designs, this is not necessary and some patterns are used. Common centroid and interdigitization are the most important ones.

Common centroid technique is the placement of blocks such that they have a common center. It is most famously known for its use in current mirrors layout. Interdigitization is the placement of blocks such that they are alternating. It is most famously used in differential pairs. Figure 1.3 shows two cases of a current mirror and differential pair laid out using those techniques. It is worthy to notice that dummies (denoted as X in the figure) are used to balance the effects on the lateral transistors. Dummies may be even used to shield all around the blocks in smaller and more sensitive technologies. More on the analog layout concept can be in the appendix.

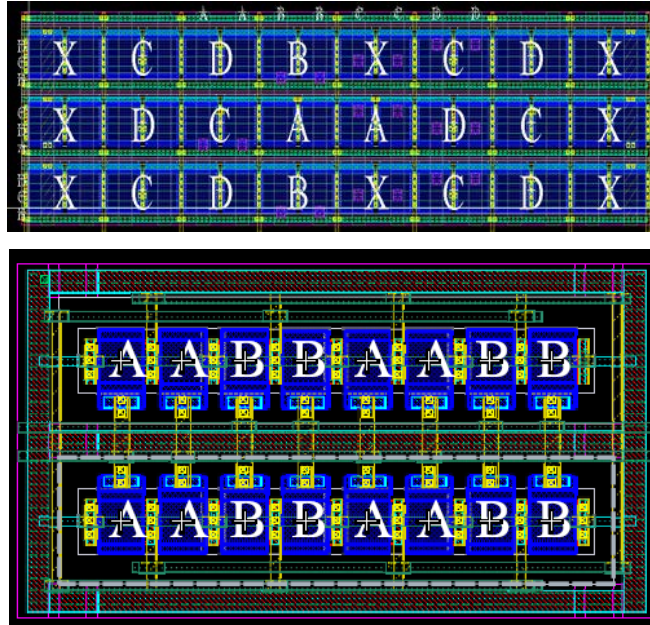


Figure 1.3: (Upper) current mirror layout using common centroid technique & (bottom) differential pair layout using interdigitization technique

1.3 Motivation for Analog Design Automation

Analog ICs shipments are currently representing more than 50% of total IC shipments [12]. This is mainly due to the boost in medical, automotive, LED lighting and energy industries where AMS blocks are used intensively and integrated in SoC designs. While AMS blocks generally occupy roughly only 3% of the total ship area, it is reported by designers that they take usually more than 50% of the total design cost [13]. This mainly due to the lack of effective CAD automation tools and the nature of analog blocks which are very sensitive to the surroundings, environmental effects and process variation and do not usually support reuse. In additions, analog designers are using CAD tools that are optimized for digital circuitry in the first place. Hence, the design cycle of AMS ICs is still time consuming and error prone. These difficulties makes the analog parts the main bottleneck in SoC designs. Given the current growth in AMS IC market, there is a motivating pressure on EDA companies to develop automation solutions for AMS circuits in order to facilitate this vicious designing cycle increasing both the productivity and even the quality of the produced work. All the designing cycles shown in figure 1.2 still have a great

room for innovation, improvement and enhancement in terms of automation, user interface and performance [13].

In the International Technology Road Map for Semiconductors [5], the V-Cycle of design compares between analog and digital automation level currently at hand. In the digital domain, EDA tools are fairly mature and well developed with nearly fully automated low-level design process. The main drawback is the lack of integration between high level (abstract) system behavioral simulations and low-level simulations. On the other hand, analog automation is way far from keeping the pace with technological advancements. The designing flow is still widely handcrafted almost completely. This is mainly due to the general trend in electronics in the last three decades which focused more on developing submicron technologies and building multimillions digital chips in accordance with Moore's law.

Analog design is generally more heuristic in nature, less systematic and knowledge intensive. Add this to the non-reusable nature of analog circuits, the problem becomes cumbersome. Recently, EDA companies are turning their attention to developing analog automation solutions based on their relevance and increased importance in modern architectures. It is important to incorporate the designers' visions, experience and knowledge for the success of such trials.

In particular, the layout generation of analog circuits is of importance. Although advancements in fabrication techniques leading to more dense circuits leading to better performance, they come at a great cost especially for analog circuits as layout induced parasitics start to emerge. Cross talk, substrate noise, supply noise, thermal noise are few examples. This problem becomes more intense in smaller technology nodes, for example at and below 40 nm, effects like well proximity, poly spacing, length of the oxide diffusion and oxide to oxide spacing become very limiting. At 20 nm nodes, layout parasitics may cause specifications' deviation up to 30% [5].

1.4 Analog Layout Automation

This book focuses mainly on analog layout automation especially on placement and detailed routing. Global routing will be touched upon but in lesser details. Analog design automation was studied heavily in academia for the past two decades and still a hot topic of research. Yet, no mature industrial automation tool is widely used till today. In fact, the whole designing process is still manually made. Few applications that give some automated functionalities to assist the designers do exist but still very limited in number and scope. Also, they are continuously ignored by designers who have the general misleading idea that analog design is an art and can never be automated. In addition to the fact the EDA companies did not put the appropriate time and effort in developing and promoting analog automation solutions. Hence, till now, the process remains same as it is, time consuming, error prone and limited.

In the traditional manual flow, the layout is only invoked when the design is fully finalized i.e. circuit sizing is complete. However, to increase the chance of first trial success and post-layout simulations that do meet the requirements, hence, alleviate the problem of backtracking and redesign which is time consuming and tiring, to do so, layout-aware or layout-driven automation methodologies must be developed. Without this, either a performance overdesign causing wasted power, area and money or performance underestimated design causing iterations of non-systematic and time consuming redesign will occur. This was the main reason for research on automatic sizing rules. Automatic sizing rules are fairly developed and even used in industrial tools, however, it must be related to layout generation (which is still lacking) and its parasitics can be estimated in a fast and robust manner to shorten the gap between the electrical and physical designs.

Generally, the complexity of layout generation do not stem from the large number of transistors (as it is the case in digital design) but from the numerous interactions among them. In addition to the imposed strict rules from the manufacturer which change from technology node to node. All of this makes it difficult to develop automation solution in a fast pace manner. At the cell-level, automation solutions do exist and though suffering from some problems, they are fairly developed. However, the bigger problem is how to integrate those solutions correctly on higher levels. Also, the routing solutions is still very primitive as nearly all the auto-routing feature available in most of EDA tools still results in unsatisfactory results.

To summarize the points we touched upon regarding analog layout automation solutions, the current challenges are:

- To develop a layout-driving circuit sizing automation solution: currently parametric cells are used to solve this problem but they still lack the flexibility and are not supportive to technology immigration.
- To build a stand-alone, robust and fast layout automatic generation tool: the tool must have the flexibility to be user assisted in order to benefit from the designers expertise. Also, the placement must be guided by constraints set by the routing to minimize the need for iterations between the placer and the router.

1.5 Advances to State-of-the-Art

The main innovative state-of-the-art solutions are discussed in chapter two in the thesis in greater details. The main disadvantages seen in multiple state-of-the-art solutions are:

- Avoiding misinterpretation of the matching constraints. Figure 1.4 shows sample result from work found in literature [14] respectively. As shown (labeled in red), the proposed solutions did not lay out the current mirrors correctly as done in industrial standards where folding transistors and modifying their widths, lengths and number of fingers and multipliers is very common to reach both an approximately matched circuit and area effective. To solve this problem, an expert guided matching pattern generator was developed to effectively integrate the designers expertise into the solutions offered.
- Reducing the computational cost. Though some of the proposed solutions found in literature, in particular perturbation-driven (stochastic) solutions, are very fast, they suffer from non-deterministic nature. Stochastic solutions mainly uses algorithms such as simulated annealing and genetic to explore the solution space of the placement problem. While it usually have a fast convergence time, it may lead to solutions with degraded quality. Deterministic solutions, on the other hand, such as Plantage [14] enumerate nearly most of the available solutions exploring all of them which is severely time consuming and unnecessary. In [15], solution space for a simple blocks consisting of eight

transistors could reach up to nearly four million solutions, in contrast, the proposed pattern generator used in our flow offered solutions are in the order of tens. Hence, the runtime are significantly enhanced.

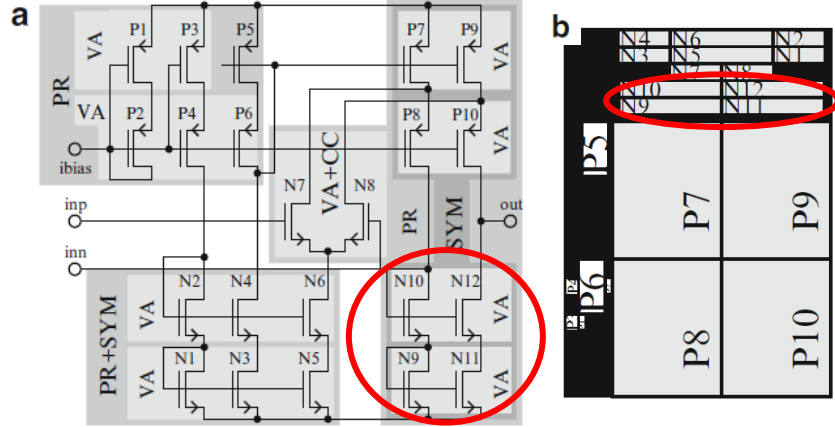


Figure 1.4: Sample result from Plantage (adopted from [14])

1.6 Conclusion

For years, AMS IC design has been a very time consuming and error prone process mainly due to the heuristic and sensitive nature of analog circuits causing analog circuit design to be the bottleneck in SoC design. Recently, the EDA vendors are shifting their attention to provide automation solution for analog designers in order to alleviate this problem and provide both faster and better analog circuit designs. In particular, layout generation has been of great interest as its rule are increasing in importance especially with lower technology nodes. Throughout this chapter, analog layout generation concepts and constraints were discussed, the goals and framework of the solutions needed were articulated and our proposed advances to the-state-of-art solutions were touched upon briefly. The next chapter will explain in a greater details the merits and limits of the currently existing state-of-the-art solutions.

References

- [1] World Semiconductor Trade Statistics (WSTS), <https://www.wsts.org/>
- [2] Moore, Gordon E. "Progress in digital integrated electronics." In *Electron Devices Meeting*, vol. 21, pp. 11-13. 1975.
- [3] Cutress, Ian. "Xilinx Announces Project Everest: The 7nm FPGA SoC Hybrid".
- [4] G. Gielen, CAD tools for embedded analogue circuits in mixed-signal integrated systems on chip. *IEEE Proc. Comput. Digit. Tech.* 152 (3), 317–332 (2005)
- [5] International Technology Roadmap for Semiconductors (ITRS) 2012 edition, <http://public.itrs.net/>
- [6] G.E. Gielen, R.A. Rutenbar, Computer-aided design of analog and mixed-signal integrated circuits. *Proc. IEEE* 88 (12), 1825–1852 (2000)
- [7] Martins, Ricardo, Nuno Lourenço, and Nuno Horta. *Analog Integrated Circuit Design Automation*. Springer, 2017.
- [8] Qian, Jessica, Satyamurthy Pullela, and Lawrence Pillage. "Modeling the" Effective capacitance" for the RC interconnect of CMOS gates." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13, no. 12 (1994): 1526-1535.
- [9] Kim, S-H., Jerry G. Fossum, and J-W. Yang. "Modeling and significance of fringe capacitance in nonclassical CMOS devices with gate–source/drain underlap." *IEEE Transactions on Electron Devices* 53, no. 9 (2006): 2143-2150.
- [10] Van Laarhoven, Peter JM, and Emile HL Aarts. "Simulated annealing." In *Simulated annealing: Theory and applications*, pp. 7-15. Springer, Dordrecht, 1987.
- [11] Davis, Lawrence. "Handbook of genetic algorithms." (1991).
- [12] IC Insights, Inc. Analog unit shipments outpacing growth of all IC product segments (2014), <http://www.icinsights.com/data/articles/documents/705.pdf>
- [13] M. Casale-Rossi, Panel: The world is going ... analog & mixed-signal! What about EDA? in *Proceedings on Design, Automation & Test in Europe (DATE)* , Mar 2014, pp. 1–5.

[14] Graeb, Helmut E., ed. Analog layout synthesis: a survey of topological approaches. Springer Science & Business Media, 2010.

[15] Balasa, Florin, Sarat C. Maruvada, and Karthik Krishnamoorthy. "On the exploration of the solution space in analog placement with symmetry constraints." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 23, no. 2 (2004): 177-191.

Chapter 2

Literature Review on the State-of-the-Art Analog Layout Automation Solutions

Several analog layout automation solutions were introduced in the literature with applications on various designs. However, in the IC industry analog layout flow is still completely handcrafted. This is essentially because of the non-competitive nature of the generated solutions and also the designers' deep belief that analog layout cannot be automated. Analog designers want to have the total control over the design process out of the belief that it is an experience-intensive and artistic process. Though that it is true that analog layout generation has a heuristic nature and sensitive to too many interactions making it hard to mathematically formulate and optimize all of them, yet, patterns emerge enabling the development of automation solutions. The literature review of the state-of-the-art solutions discussed throughout this chapter reveals that analog layout automation that was once ignored is now developing rapidly and a mature industrial-grade tool may be within our reach in the upcoming few years.

After the electrical path of an analog circuit path is completed i.e. the topology is selected and sized, the circuit must be physically laid out. It is of common practice to separate this problem into two main components, placement and routing. This chapter starts by addressing the placement problem and how it can be solved with especial care to the different floorplan representations that can be used and how this representation choice might affect the computational time. Also, the challenges and limitations found would be addressed. After that, the routing problem will be addressed giving an overview of the existing algorithms with more focus on electro-migration aware tactics and wiring symmetry constraints. Following, a section is dedicated to review existing both academic and commercial tools. Also, visions and insights to develop a layout-aware designing optimizer to correctly size the circuit given the layout-induced effects are discussed. Lastly, an overview of our implementation choices and how we benefited from the previous existing work to build upon and improve is presented.

2.1 Placement

For a placement tool to generate a floorplan that meets the industrial IC standard performance, it should be able to minimize the area, taking into accounts parasitic effects such as process variations and various on-die gradients i.e. thermal and stress gradients and allow for appropriate suitable routing scheme. All those requirements must be met simultaneously for a layout to be admissible making the automation task very complex. Those requirements can be translated in terms of certain constraints such as symmetry, matching and proximity in addition to the various matching implementation schemes i.e. common centroid, common gate and interdigitization allied with a multitude of transistor folding possibilities. Also, the DRCs imposed by the manufacturer must be followed, given in mind that for a tool to be technology supportive, it must have the capability to understand and absorb different DRC rules. It must be also fully understood how the placement would affect the routing scheme and its quality as once a floorplan is set, the upper limit of routing quality and parasitics effects are set as well.

2.1.1 Analog Topological Constraints

As mentioned, the topological constraints for analog placement are device matching, symmetry and proximity [1]. Those constraints are demonstrated in figure 2.1. Symmetry restricts devices to be mirrored around a certain axis. This could be beneficial in cases such as differential pair where the placement of the differential pair and the connected devices to it must be symmetric not only to balance the effects on the differential pair transistors but also to allow for symmetrical routing. Matching, on the other hand, is a pattern-like placement. Common matching tactics are common gate, common centroid and interdigitization. Common centroid is usually used for current mirrors while interdigitization is used generally in differential pairs. Those patterns decrease the susceptibility of circuits to process variations but usually not very effective in aggressive non-linear effects [2-3]. Lastly, the proximity limits the positions of the devices either they can be close to a certain distance, share a common substrate/well region, surrounded by a guard ring or even be

placed closely to symmetrical/matched groups. By following proximity constraints, several degrading effects are lowered in effect such as coupling and large mismatches [1, 4]. Equation 2.1 shows an empirically driven mismatch function which is proportional to distance, D_x , (proximity).

$$\sigma^2(\Delta P) = \frac{A_p^2}{WL} + S_p^2 D_x^2 \quad (2.1)$$

where A_p & S_p are respectively area and separation proportionality constants, D_x separation distance, $\sigma^2(\Delta p)$ is the mismatch induced deviation, W is the transistor width and L is its length.

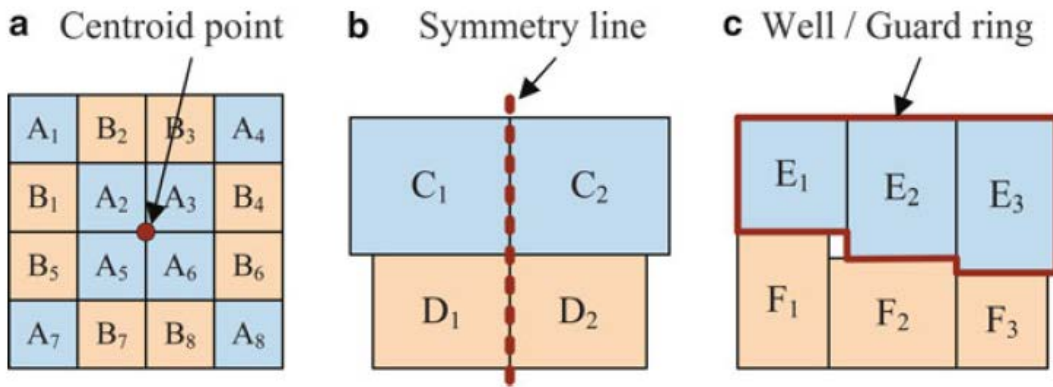


Figure 2.1: Common topological constraints. A) Matching pattern B) Symmetry C) Proximity (adopted from [1])

2.1.2 Floorplan Representations

Floorplan representation is one of the most crucial choices when developing an automated placer as it affects the algorithm structure and hence affects both how the relations among blocks are defined and the computational cost to explore the solution space. Generally, the floorplan representations falls into two categories, absolute representation i.e. the use of absolute coordinates and topological representations i.e. the relative positioning of blocks to each other's. Topological representation can be further categorized into slicing and non-slicing representations.

Manipulating the floorplan representation to achieve the optimum floorplan is either done deterministically or stochastically. Stochastic solvers are more common as they are superiorly faster. However, they do not guarantee obtaining the best results and usually have a complex cost

function that might lead to ignoring/underweight/overweight some factors. Stochastic solvers famously use customized versions of simulated annealing and genetic algorithms. This section will focus more on stochastic solution methods. An upcoming section would be dedicated to discuss the most prominent deterministic solution –to the authors’ knowledge- Plantage.

2.1.2.1 Absolute Representation

In the absolute system, every block is identified by its absolute co-ordinates. Usually, a stochastic optimizer has the freedom to move the cells in every possible placement. This representation is much easier to develop than topological representations, yet, the description of the interactions between the block is hard to describe. A cost function is used to guide the optimizer with several goals i.e. eliminate illegal overlap, minimize the area, minimize the net length, have a certain aspect ratio...Etc. Due to the huge search space that need to be explored and the non-deterministic nature of the searching process, the final placement may still have overlapping or even do not meet the required specifications. This usually necessitates the use of an additional post-processing step to fix this problem.

One of the first uses of absolute coordinates representation applied to analog circuits was in KOAN/ANAGRAM II [5] where a simulated annealing kernel [6] was used. The used cost function is:

$$\min f(x) = \alpha_1 f_1 + \alpha_2 f_2(x) + \sum_i \alpha_i f_{i(x)} \quad (2.2)$$

where x is the co-ordinates of each block, σ is the weight and $f_1(x)$ and $f_i(x)$ are the fundamental primary goals that needs to be met. $f_1(x)$ represented the overlapping which needs to be driven to zero and $f_i(x)$ included all the other objects i.e. area, net length, aspect ratio, proximity...Etc.

LAYLA [7], ALDAC [8] and PUPPY-A [9] also used the same cost function formula with variations on $f_i(x)$ such as net length estimation methods i.e. half perimeter, pseudo Steiner tree ...etc. [9] and performance degradation prediction tactics [7, 9]. The σ is usually determined experimentally or adopt dynamically within the process. If dynamically adopted, σ is set at the

beginning such that area and wire length dominates the cost function, then, later (at low temperature) the kernel shifts the importance towards illegal overlap and symmetry [9]. Yet, the tuning of σ is not always straight forward and may lead to undesired results. Generally, the customization of the cost function is one of the main demerits of the absolute system representation.

Other attempts to use genetic algorithm rather than simulated annealing were made [10-12]. Generally, genetic algorithms leads initially to faster convergence rates, yet, the search efficiency degrades rapidly due to the incapability of the algorithm to accept unfavorable solutions leading premature suboptimal solution. Another approach was to use genetic-based algorithm whose mutations are controlled by simulated annealing based algorithm [13]. In that approach, unfeasible solutions were avoided by means of post-processing technique called deterministic cell slide where after each mutation, the absolute placement is transformed to a relative topological one, hence, solving the overlaps and forcing symmetry constraints. The main adversary of this method was the need to transform the absolute position system to a topological one.

2.1.2.2 Topological Representations

In the topological representations, the stochastic optimizer no longer can move the block in an explicitly manner but perturb the representation in a manner which changes the encoding of the representation and hence its decoding (placement). This necessitates the need to fully understand the encoding (packing) of the floorplan and how this encoding can absorb and support the constraints. The solution space can be greatly reduced if configurations that cannot support the required configuration are skipped (unfeasible placements) and only the feasible regions are only explored. However, depending on the representations used, the optimal solution can be or not be found within the search space. This stems from the fact that not all the topological representations support the encoding of all the required constraints. Furthermore, since the placement information is encoded and cannot be directly driven, all the solutions must be decoded (usually using a linear solver) to check for the most optimal solution and dismiss sub-optimal solutions.

2.1.2.2.1 Slicing Representations

In the slicing representation, the cells are organized by a set of slices that bisect the layout either vertically or horizontally. The representation could be either seen as a tree of a polish expression. Figure 2.2 shows a layout and how it is represented by slicing tree where “h” stands for a horizontal cut and “v” a vertical cut. Due to the fact that not all the layout can be perfectly sliced, the solution could be degraded in terms of area. This is in fact is a very high possibility in analog layout where blocks generally have contrasting aspect ratios. This problem can be alleviated if the slicing was followed by compaction [14].

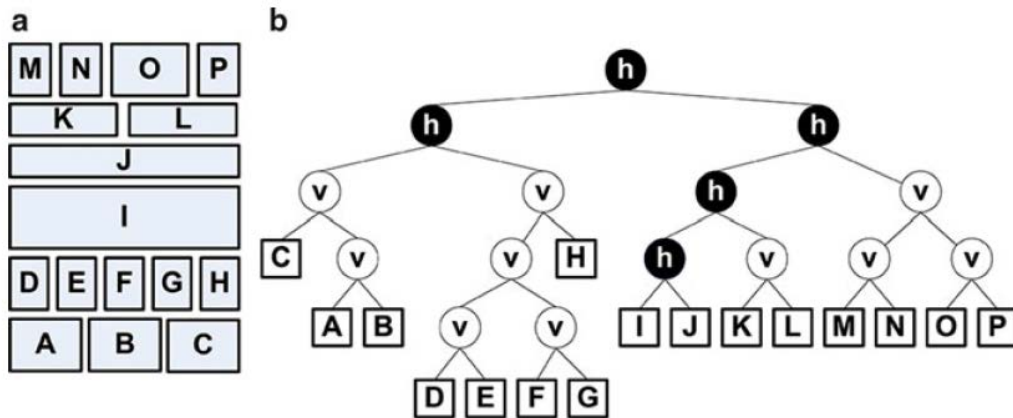


Figure 2.2: Layout example and its representation as slicing tree (adopted from [1])

Slicing representation has been used for layout migration [15], fixed outline floorplan [16] and symmetric feasibility conditions [17]. However, due to the draws in the slicing representation, other non-slicing representations were used where area degradation due to the existence of non-sliced layouts is no longer a problem.

2.1.2.2.1 Non-Slicing Representations

Figure 2.3 shows a famous example of a non-slicing floorplan known as “wheel”. Non-slicing representations are more generic and suitable to the nature of the analog layout problem.

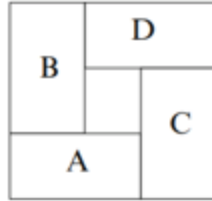


Figure 2.3: Non-slicing topology example known as wheel (adopted from [1])

There are several non-slicing representations such as:

- Sequence-pair: encodes the “left-right” and “up-down” relations between blocks [18]. The solution space can be explored effectively and can be decoded into any required placement. Symmetry constraints can be handled by the sequence-pair representations. The packing computational cost are in the order in $O(n^2)$ [19-20]. In [21], this cost was relatively enhanced to $O(G.n.\log(\log(n)))$ where G is the number of symmetry groups.
- Bounded-sliceline grid (BSG): uses a meta-grid structure where relations are uniquely defined using “right-of” and “above” relations for each block [22]. BSC has an $O(n^2)$ packing complexity but much more intuitive than sequence-pair, yet, there is no prove that it can support all the constraints especially the symmetry. Hence, it may be incapable of representing the optimal solution.
- The ordered tree (O-tree): is an extended version of the slicing tree with even lesser complexity than the original slicing tree. It reduces the redundancies found in sequence-pair and BSG. In addition, it requires fewer space complexity than them. The complexity of transforming an O-tree (without symmetries) is of $O(n)$ complexity. The inclusion of symmetry is introduced in [23].
- B*-Tree: is a binary graphy representation with $O(n.\log(n))$ packing complexity and do not require additional constraint graphs for cost computation [24]. The root of the B*tree corresponds to the bottom left block. B*tree have many mature methods to reduce the set of symmetric feasible sets using segements trees [25], red-black interval trees [26] and skip lists [27].

Figure 2.4 shows an example of the representation of a topology in sequence-pair, O-tree and B*tree encodings.

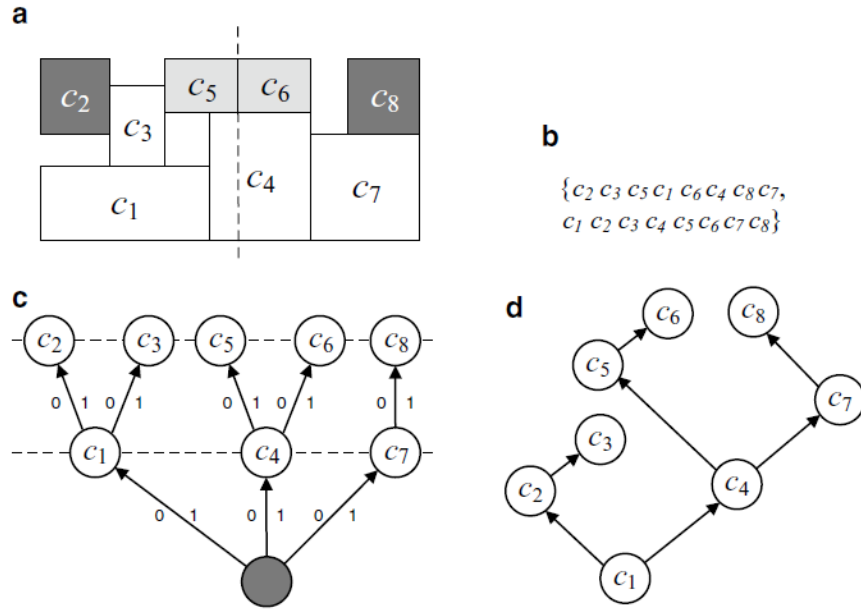


Figure 2.4: a) Layout topology example b) Its representation in sequence-pair c) O-tree d) B*tree

- Transitive closure graph-based (TCG): combines all the merits of the sequence-pair, BSG and B*tree with unique feasible packing with no additional constraints graphs needed for cost evaluation [28]. Modified in [29] to TCG-S, TCG-S is a TCG combined with a sequence-pair to inherit the fast packing properties of the sequence-pair without affecting other aspects in the TCG. Symmetry feasible conditions were developed for both TCG and TCG-S in [30] and [31] respectively.
- Recently, in [32-34], the concept of symmetry islands was introduced which is the only representation that deals with proximity constraints effectively. This is done by keeping symmetry groups elements (elements that share the same axis of symmetry) close to each other's. This concept is implemented using B*tree transforming it to what is called "automatically symmetric feasible (ASF) B*tree". The principle idea is using the benefits of the hierarchy found in analog circuits building hierarchal B*trees (HB*trees) to guide the whole process of placement. The same concept was applied also to a slicing tree in [35]. This framework provides a holistic representation capable of handling all the constraints of modern analog circuits.

In table A.1, summary of the advantages, drawbacks, complexities of each representation discussed.

2.1.3 Modern analog placement challenges and schemes

Various methodologies have been used to utilize the representations discussed above and how to integrate them in a placement generation tool. Modern tools are now faced with new challenges that re-shaped the placement problem beyond the primary objectives i.e. area and wire length minimization. Thermal effects, sensitivity of signal-paths, layout-induced parasitics are few among those challenges.

2.1.3.1 Pareto Front of Placement Solutions

Plantage [4] is a fully deterministic placement solution. It is mainly based on the fact that analog circuits tend to have a hierarchical structure. First, the repeating building blocks in the circuit are identified, then, a hierarchical tree is built based on the constraints (ordered by their importance) imposed by those building blocks i.e. current mirrors require common centroid matching, differential pairs require symmetry ..Etc. After that, the whole possibilities of placements for each identified block is enumerated. Lastly, they are combined using a linear solver followed by a shape function. All the enumerated solutions are represented by the shape function and the pareto front of placement solutions i.e. solutions with the minimum area are given to the designer to choose from them based on the aspect ratio. It is important to notice that since the whole process was guided by the hierarchical tree (HB*tree), all the solutions meet the constraints i.e. symmetry, matching and proximity while in the same time offers area minimization. One of the main advantages of the work of Plantage is the shape function used and the hierarchical building of the constraints. The shape function called “enhanced shape function” is much better than the traditional shape function as it offers not only vertical and horizontal addition but also allows for orientation changes. Figure 2.5 shows this property. It is worthy to notice that the enumeration

happened only at the lower level (building blocks) due to the fact that it would be very computationally expensive to enumeration the whole levels of hierarchy. This is mitigated by the use of the shape function.

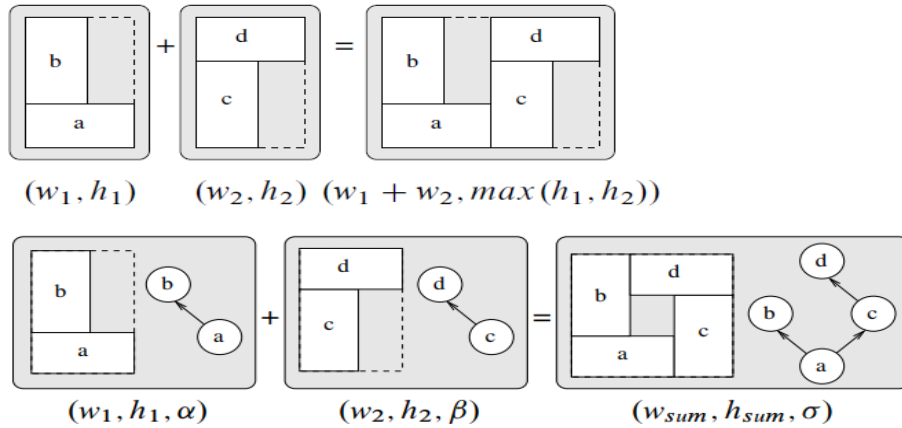


Figure 2.5: Comparison between traditional shape function (above) and enhanced shape function (below).

2.1.3.2 Thermal-Driven Placement

In the work of [36-38], a thermal-driven placer was proposed. It optimizes power and non-power placement where power devices are defined as devices consuming much higher power than their counterpart, non-power devices. This is made in attempt to annihilate thermal mismatches. This is especially important as power devices generated thermal impact might degrade the electrical characteristics of other thermally-sensitive modules. The algorithm works towards the goal of even thermal distribution for the heat chip.

2.1.3.3 Current-Driven Placement

As mentioned earlier, the placement set an upper limit of the best attainable routing and hence, parasitics effects too. This necessitates the need for placement solutions aware –in advance– of the considerations of current flow and density. Smooth routing (current flow) both reduce mismatches and parasitics, also, taking the current density into consideration are of essence to overcome problems of IR-drop and electromigration.

In [39], the first approach to consider current-flow aware placer was made by limiting the placement the cells in order to ensure the same current flow direction throughout the whole layout. Later in [16], the deterministic algorithm DeFer operated over a hierarchal slicing-tree providing solutions that has a single monotonic current path. The wire length is then optimized using linear solver when all the monotonic current path constraints are met. In [40], both the current flow and density were addressed by incorporating both the devices and their interconnects in an enhanced HB*tree. Then, a constructive dynamic tool simultaneously place and route the modules and reserving spaces for routing between cells.

2.1.4 Stochastic Optimizer

As mentioned earlier, the simulated annealing is best fitted optimizer for the analog placement problem due to its ability to explore the solution space effectively. Simulated annealing gets its name from its stochastic nature of searching point-to-point solutions similar to annealing process in solids. One of the main aspects of simulated annealing algorithm is the probability to accept explored solutions which is governed by equation 2.3 [6]:

$$p(f(u), f(v)) = \frac{1}{1 + e^{[(f(u)-f(v))/(T*f(u))]} \quad (2.3)$$

where $f(u)$ and $f(v)$ are the relative “goodness” of the current and next solutions. According to the parameter T , this probability could be sharp or smooth. T is usually scheduled as to decrease exponentially as a function of time. This means as time advances, the probability to accept new solutions get lower. Equation 2.4 shows T modelled as an exponential function of time:

$$T(t) = T_{max} * e^{-R*\left(\frac{t}{k}\right)} \quad (2.4)$$

where R is the temperature decreasing rate, k is a scale factor for the iteration counter t and T_{\max} is the initial temperature. This way simulated annealing could be tuned to effectively explore the search space and even tolerate unfavorable solution in the sake of pursuing the optimal one.

2.1.5 Commercial Solutions

Some automation features started to emerge into commercial EDA tools. Tanner EDA [41], HiPer DevGen acquired by Mentor Graphics offers generators that automatically creates parametrized cells for common analog building blocks i.e. current mirrors and differential pairs. The generated cells are very efficient and similar to those made manually. Options are given to the designers to control some of the features in the generated cells. For example, for differential pairs there are options like matching pattern, optimized parasitics, dummies and guard rings additions, antenna effect diode...etc. While in current mirrors there were the options to different electric-current strengths, matching patterns, dummy addition, diffusion sharing, adjustments to well proximity effect ...etc. For each technology nodes, the DRCs fed into the generators must be changed.

Synopsys HelixTM [42] is an automated placement tool with user-friendly graphical user interface (GUI). The designer first introduces the system hierarchy and its sub-blocks independently. Then, the tool can provide the designer with estimated area and parasitics so that the designer can focus on higher level optimization. For the automatic placement, the designer provide the required constraints and the tool provides the possible minimum-area placements that satisfy those constraints. Those solutions are generated deterministically.

2.2 Routing

As mentioned earlier, in the traditional layout generation flow, routing is only triggered after the placement is complete. With that stated, routing techniques and state-of-the-art routers

are reviewed. It is worthy to mention that most of the used auto-routers till recently have been greatly discredited by designers as they do not provide the required quality.

2.2.1 From Netlist to Pathfinder

The netlist contains the devices that needs to be connected but not the path these interconnects should follow. This problem is usually called a classical Steiner minimal tree (SMT) problem. In that problem, a set of Steiner trees must be found in order to minimize the whole wire lengths of the interconnects. This problem is usually thought as rectilinear Steiner tree (RMST) as the routes in the analog circuits are usually rectangles (edges are defined by rectangular vertical and horizontal segments). This problem, however, is still NP-complete [43].

First, a RMST router is used to for wire length estimation [44]. Then, when the terminal-to-terminal connectivity is found, a traditional deterministic pathfinder algorithm is used. Pathfinder algorithms are variation of the classic maze algorithm [44-46] which is the most common approach used in automated routing today due to their deterministic nature. Other algorithms includes line-expansion techniques [47]. The basic concept of those approaches is to make a connection between two terminals avoiding obstacles (other devices/wires). This is done usually using a grid-like of tile-based representations to avoid any DRC violations. To avoid conflicting nets, heuristics for importance, backtracking and re-routing are used.

Other approach to the routing problem is the use of template-based techniques [48, 49] where a user-assisted generation of the routes is made. This approach is slower but usually better in results.

2.2.2 Electromigration and IR-drop

Electromigration and IR-drop are of the main non-idealities in routing. Electromigration is the molecular migration of atoms due to electron movements that leads eventually to breaking the

metal connects. IR-drop, on the other hand, is the change of the net voltage assigned to an element due to interconnect resistance which in turn affects the performance and may even the functionality of the circuit [50, 51].

In 1969, J. R. Black modeled empirically the time required for an interconnect failure based on interconnect area, current density and temperature [52]. Equation 2.5 shows the modeled developed for aluminum conductors used in early days of IC industry.

$$\text{Median Time to Failure (in hours)} = \frac{A}{J^n} * \exp\left(-\frac{\emptyset}{k.T}\right) \quad (2.5)$$

where A is a constant that a factor in cross sectional area of the wire, J is the current density going through the wire, \emptyset is the electron activation energy, k is Boltzmann constant, n is a scaling factor and T is the temperature. Observing equation 2.5, it is noticed that only two parameters are within the control of the designer: I. The wire width and hence the cross-sectional area, the smaller the width, the faster the failure. II. Temperature which might be indirectly controlled by allocating changing the placement of power/thermally-sensitive devices and routes. Though the work of J. R. Black is only limited to the used technology back then and is vastly outdated, those concepts are still valid in today's technology.

2.2.3 Electromigration-Aware Approaches

To solve this problem, generally, the wire planning is done first where a tree with flow of current between terminals is built. Kirchhoff's current law must be valid at each terminal. The terminals of the same net must be connected with minimum allowed wiring area. After that, the routing paths are rectilinearized.

In [60], semi-automatic technique called "three-point Steinerization" was used constructing a Steiner tree for a set of single-port sources and sinks with different current densities adding up the nearest terminal to the current sub-tree sequentially. This solution might need manual assistance to move some obstacles or re-allocate some the routes.

In [53], a greedy algorithm is used with the primary goal to maximize the interconnect area gain. In [54], an improved version with better obstacle avoidance characteristics. In [55], a linear solver is used to aid the construction of SMT by handling multiple currents, yet, it has a very expensive computational cost. In [56], a wiring topology is used, especially for high-current circuits, where the problem is broken own to small clusters that are routed independently in an exhaustive manner. In [57, 58], the NP-hard planning problem is converted to a P-problem using the greedy choice property. This work, unlike the previous methods, considers all the sources in the planning step. Finally, in [59], channel space restriction is taken into consideration.

2.2.4 Wiring Symmetry

Wiring symmetry is of particular importance in the routing of differential circuits. The problem are decomposed into two section, first, symmetrical placement and symmetrical routing. With those two conditions achieved, the circuit can be truly differential. The works of [1] are few examples of literature found on the subject.

The most known commercial solutions are Mentor Graphics' IRoute and ARouter and Cadence Virtuoso Chip Assembly Router.

2.3 Complete Layout Generation tools

A variety of analog layout generation and some tools are reviewed next. First attempts for analog layout automation was known as procedural generation. Procedural module generation used parametric cell which is commonly known as pCell. The parametric cell concept is coding the entire layout of a circuit in a software tool. Hence, it is fully developed by the designer. A tool called ALSYN uses fast procedure algorithms. However, this method lacks the flexibility to adapt to wide changes. Hence, the cost of making a new design is high and technology migration may need redesign.

The second type is template based. It uses a pre-defined template that has the relative position and interconnection of devices. A tool called IPRAIL, which stands for Intellectual Property Reuse-based Analog IC Layout, extracts the information from already made layout automatically for retargeting purposes. Layout retargeting means using an existing layout to generate a new layout. Hence, this method can be used for technology migration, update specifications, or optimizing the old design. A tool is introduced by Po-Hsun Wu et al.[] to generate a new layout by integrating existing design expertise.

The third type is optimization based. These approaches synthesize the layout by means optimization techniques based on some cost functions. This type is easier to implement compared to procedural and template-based approaches.

Now, let us consider commercial solutions. Virtuoso® Layout Suite Family made the creation and navigation in a complex design. This method manages multiple abstraction levels for device, cell, block, and chip levels. Moreover, it contains different level of assistance. Basic design creation assisted correct-by-construction wire-editing functionalities to ensure real time process. Hierarchical designs intent for schematic editor. Synopsys'® Galaxy Custom Designer LE also offers a set of automation functionalities for layout generation. It has an automatic guard ring generator which creates guard rings in real-time and it is easily edited. Furthermore, user assisted applications as auto connect features, interactive bus routing with different via pattern choices, align assist displays with interactive alignment, commands to specify locations to create bridges or a tunnel for routing, these frameworks adds several functionalities to speed-up layout design.

The gap between electrical and physical design needs to be closed by ensuring that the post layout performance is guaranteed in the presence of layout parasitic. Iterations between the physical design and the electrical simulation is ineffective as it is a time-consuming process. The proposed solution is using a layout-induced effects in the circuit sizing phase to overlap electrical and physical design phases. Hence, parasitic-aware, layout-aware or layout-driven sizing should be made. device parasitic effects are modeled by linear regression obtained by sampling the design space and with means of procedural generator it produces the layout for each point, where each solution is aware of its specific layout induced effect. Hence, the effects can be expected without actually generating the layout in real-time. Moreover, the parasitic effects are modeled in the

circuit equations. Hence, a deterministic nonlinear optimization algorithm is used to determine the device sizes.

The following table 2.1 (adopted from [63]) shows analog layout aware sizing tools.

Table 2.1 Comparison between state-of-the-art works on layout-aware sizing with special emphasis to layout-related data included in-loop

Work	Layout Generator		Layout-related data included							Observations
	Placer	Router	Geometric	Parasitics				Resistances		
				Bulk	Intra-device	Inter-device	Interconnect			
Vancor. [99]	Procedural generator		✓	✗	✓	✗	✓	✓	✓	1/2-D analytical-geometrical modeling of the critical nets; geometric data with equations
Pradhan [94]	Procedural generator (design space sampled only)		✓	✓	✓	✗	✗	✗	✗	Analytical models for bulk, device and interconnect
Liao [96]	Template-based/user-assisted	Channel router	✓	✗	✓	✓	✓	✓	✓	Analytical models for area and interconnect
Youssef [97]	Device-level procedural generator		✓	✓	✓	✗	✗	✗	✗	Modeling of the stress effects of the devices
Ranjan [95]	Procedural generator (design space sampled only)		✓	✗	✓	✓	✓	✓	✓	Area and interconnect using external extractor
Habal [103]	Enumeration of all possible floorplans	Exhaustive setup for Cadence Chip Assembly Router [®]	✓	✓	✓	✓	✓	✓	✓	Complete extraction using Cadence Assura [®]
Lopez [100, 101]	Coded slicing-tree	Template-based	✓	✗	✓	✓	✓	✓	✗	3-D analytical-geometrical
Berkol [102]	Layout description script		✓	✓	✓	✓	✓	✓	✓	Complete extraction using external extractor
This work, AIDA-L	Template-based Placer with multiple B*-trees	Automatic electromigration-aware WT and global routing in-loop	✓	✓	✓	✓	✓	✓	✓	2.5-D modeling of the devices and routing that operates over non-detailed routing

The following diagram shows a Chronological representation of analog layout generators.

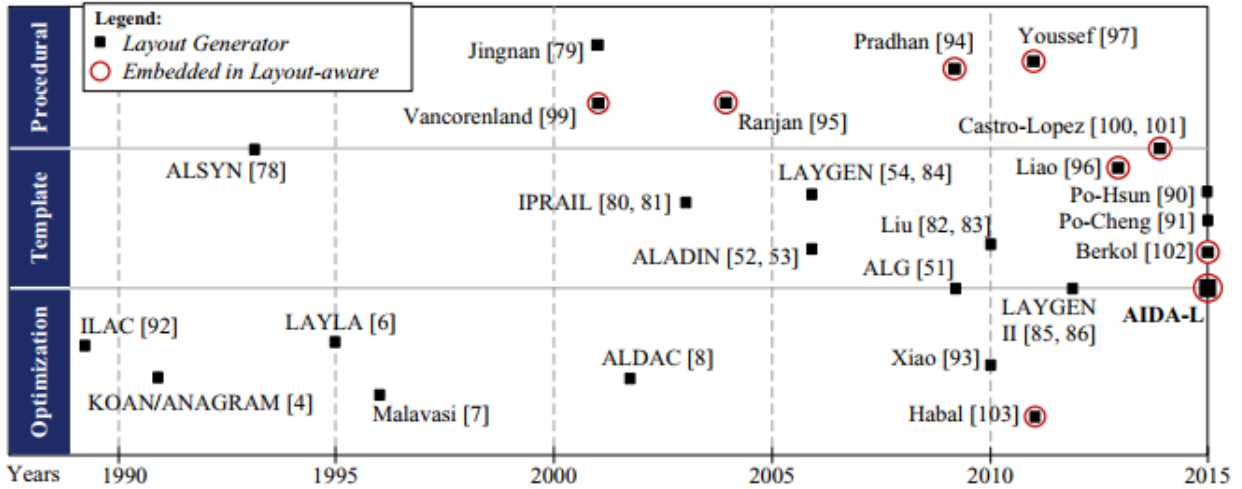


Figure 2.6: Chronological representation of analog layout generators (adopted from [63])

The following table 2.2 (adopted from [63]) shows the optimization algorithms used in analog placement.

Table 2.2 Optimization algorithms used in analog placement automation tools

Algorithm/representation	Simulated annealing	Genetic algorithm	GA and SA	Linear/dynamic programming
Absolute	[4, 6–8]	[10–12]	[13]	[46]
Slicing tree	[14, 17, 45, 92]	X	X	[15–17, 40, 47]
Sequence-pair	[20, 21, 23, 24]	X	X	[22]
BSG	[25]	X	X	X
O-tree	[27]	X	X	[26]
B*-tree	[28–32, 37–39]	X	X	[2]
TCG/TCG-S	[33–35]	X	X	X

References

- [1] Graeb, Helmut E., ed. Analog layout synthesis: a survey of topological approaches. Springer Science & Business Media, 2010.
- [2] Borisov, Vadim, Kerstin Langner, Jürgen Scheible, and Benjamin Prautsch. "A novel approach for automatic common-centroid pattern generation." In 2017 14th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), pp. 1-4. IEEE, 2017.
- [3] El Din, Islam Nashaat Salah, Mohamed Dessouky, and Hazem Said. "Optimally matched current mirror layout pattern generation using genetic optimization." In 2016 28th International Conference on microelectronics (ICM), pp. 145-148. IEEE, 2016.
- [4] Strasser, Martin, Michael Eick, Helmut Gräß, Ulf Schlichtmann, and Frank M. Johannes. "Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions." In Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, pp. 306-313. IEEE Press, 2008.
- [5] J. Cohn, J. Garrod, R.A. Rutenbar, L. Carley, KOAN/ANAGRAM II: new tools for device-level analog placement and routing. IEEE J. Solid State Circuits 26(3), 330–342 (1991)
- [6] B. Suman, P. Kumar, A survey of simulated annealing as a tool for single and multiobjective optimization. J. Oper. Res. Soc. 57, 1143–1160 (2006)
- [7] K. Lampaert, G. Gielen, W. Sansen, A performance-driven placement tool for analog integrated circuits. IEEE J. Solid State Circuits 30(7), 773–780 (1995) 2 State-of-the-Art on Analog Layout Automation 37
- [8] E. Malavasi, E. Charbon, E. Felt, A. Sangiovanni-Vincentelli, Automation of IC layout with analog constraints. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 15(8), 923–942 (1996)
- [9] P. Khademsameni, M. Syrzycki, A tool for automated analog CMOS layout module generation and placement. IEEE Can. Conf. Elect. Comput. Eng. 1, 416–421 (2002)
- [10] H. Chan, P. Mazumder, K. Shahookar, Macro-cell and module placement by genetic adaptive search with bitmap-represented chromosome. Integr. VLSI J. 12(1), 49–77 (1991)

- [11] V. Schnecke, O. Vornberger, Hybrid genetic algorithms for constrained placement problems. *IEEE Trans. Evol. Comput.* 1(4), 266–277 (1997)
- [12] B.H. Gwee, M.H. Lim, A GA with heuristic-based decoder for IC floorplanning. *Integr. VLSI J.* 28(2), 157–172 (1999)
- [13] L. Zhang, R. Raut, Y. Jiang, U. Kleine, Placement algorithm in analog-layout designs. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 25(10), 1889–1903 (2006)
- [14] L. Minghorng, D.G. Wong, Slicing tree is a complete floorplan representation, in *proceedings on Design, Automation and Test in Europe (DATE)*, Mar 2001, pp. 228–232
- [15] Y.-P. Weng, H.-M. Chen, T.-C. Chen, P.-C. Pan, C.-H. Chen, W.-Z. Chen, Fast analog layout prototyping for nanometer design migration, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2011, pp. 517–522
- [16] J.Z. Yan, C. Chu, DeFer: deferred decision making enabled fixed-outline floorplanning algorithm. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 29(3), 367–381 (2010)
- [17] M.P. Lin, B.-H. Chiang, J.-C. Chang, Y.-C. Wu, R.-G. Chang, S.-Y. Lee, Augmenting slicing trees for analog placement, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sept 2012, pp. 57–60
- [18] H. Murata, K. Fujiyoshi, S. Nakatake, Y. Kajitani, VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 15(12), 1518–1524 (1996)
- [19] F. Balasa, K. Lampaert, Symmetry within the sequence-pair representation in the context of placement for analog design. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 19(7), 721–731 (2000)
- [20] S. Koda, C. Kodama, K. Fujiyoshi, Linear programming-based cell placement with symmetry constraints for analog IC layout. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 26(4), 659–668 (2007)

- [21] K. Krishnamoorthy, S. Maruvada, F. Balasa, Topological placement with multiple symmetry groups of devices for analog layout design, in Proceedings IEEE International Symposium on Circuits and Systems, May 2007, pp. 2032–2035
- [22] S. Nakatake, K. Fujiyoshi, H. Murata, Y. Kajitani, Module packing based on the BSG-structure and IC layout applications. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 17(6), 519–530 (1998)
- [23] Y. Pang, F. Balasa, K. Lampaert, C.-K. Cheng, Block placement with symmetry constraints based on the o-tree non-slicing representation, in Proceedings ACM/IEEE Design Automation Conference, 2000, pp. 464–467
- [24] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, S.-W. Wu, B*-trees: A new representation for nonslicing floorplans, in Proceedings of the 37th ACM/IEEE Design Automation Conference (DAC), 2000, pp. 458–463
- [25] F. Balasa, S.C. Maruvada, K. Krishnamoorthy, On the exploration of the solution space in analog placement with symmetry constraints. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 23(2), 177–191 (2004)
- [26] F. Balasa, S. Maruvada, K. Krishnamoorthy, Efficient solution space exploration based on segment trees in analog placement with symmetry constraints, in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Nov 2002, pp. 497–502
- [27] F. Balasa, S.C. Maruvada, K. Krishnamoorthy, Using red–black interval trees in device-level analog placement with symmetry constraints, in Proceedings of the Asian and South Pacific—Design Automation Conference (ASP-DAC), Jan 2003, pp. 777–782
- [28] S. Maruvada, A. Berkman, K. Krishnamoorthy, F. Balasa, Deterministic skip lists in analog topological placement, in Proceedings 6th International Conference On ASIC (ASICON), Oct 2005, vol. 2, pp. 834–837
- [29] L. Jai-Ming, C. Yao-Wen, TCG: a transitive closure graph-based representation for non-slicing floorplans, in Proceedings of the 38th ACM/IEEE Design Automation Conference (DAC), 2001, pp. 764–769

- [30] L. Lin, Y.-W. Chang, TCG-S orthogonal coupling of P-admissible representations for general floorplans. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 23(5), 968–980 (2004)
- [31] L. Zhang, C.-J. Shi, Y. Jiang, Symmetry-aware placement with transitive closure graphs for analog layout design, in *Proceedings of the IEEE/ACM Asia South Pacific Design Automation Conference*, Mar 2008, pp. 180–185
- [32] J.-M. Lin, G.-M. Wu, Y.-W. Chang, J.-H. Chuang, Placement with symmetry constraints for analog layout design using TCG-S, in *Proceedings of the IEEE/ACM Asia South Pacific Design Automation Conference*, Jan 2005, vol. 2, pp. 1135–1138
- [33] P.-H. Lin, S.-C. Lin, Analog placement based on novel symmetry-island formulation, in *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC)*, June 2007, pp. 465–470
- [34] P.-H. Lin, S.-C. Lin, Analog placement based on hierarchical module clustering, in *Proceedings of the 45th ACM/IEEE Design Automation Conference (DAC)*, June 2008, pp. 50–55
- [35] P.-H. Lin, Y.-W. Chang, S.-C. Lin, Analog placement based on symmetry-island formulation. *IEEE Trans. Comput. Aided Des.* 28(6), 791–804 (2009)
- [36] P.-H. Wu, M. Lin, T.-C. Chen, C.-F. Yeh, T.-Y. Ho, B.-D. Liu, Exploring feasibilities of symmetry islands and monotonic current paths in slicing trees for analog placement. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 33(6), 879–892 (2014)
- [37] P.-H. Lin, H. Zhang, M. Wong, Y.-W. Chang, Thermal-driven analog placement considering device matching, in *Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC)*, July 2009, pp. 593–598
- [38] P.-H. Lin, Recent research in analog placement considering thermal gradient, in 20th *European Conference on Circuit Theory and Design (ECCTD)*, Aug 2011, pp. 349–352
- [39] P.-H. Lin, H. Zhang, M. Wong, Y. Chang, Thermal-driven analog placement considering device matching. *IEEE Trans. Comput. Aided Des.* 30(3), 325–336 (2011)

- [40] D. Long, X. Hong, S. Dong, Signal-path driven partition and placement for analog circuit, in Asia and South Pacific Conference on Design Automation (ASP-DAC), Jan 2006, pp. 694–699
- [41] H.-C. Ou, H.-C. Chien, Y.W. Chang, Simultaneously analog placement and routing with current flow and current density considerations, in Proceedings of the 50th ACM/IEEE Design Automation Conference (DAC), June 2013, pp. 1–6
- [42] Mentor Graphics' Tanner EDA (2016), Tanner AMS and MEMS Flows.
<https://www.mentor.com/tannereda>
- [43] Synopsys (2016), Synopsys. <http://www.synopsys.com>. Accessed 20 May 2016
- [44] G. Robins, A. Zelikovsky, Minimum Steiner Tree Construction, in The Handbook of Algorithms for VLSI Physical Design Automation, ed. by C.J. Alpert, D.P. Mehta, S.S. Sapatnekar (CRC Press, Boca Raton, 2009), pp. 487–508
- [45] Y. Yilmaz, G. Dundar, Analog layout generator for CMOS circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 28(1), 32–45 (2009)
- [46] L. Zhang, U. Kleine, Y. Jiang, An automated design tool for analog layouts. IEEE Trans. Very Large Scale Integr. VLSI Syst. 14(8), 881–894 (2006)
- [47] L. Zhang, Y. Jiang, Global-routing driven placement strategy in analog VLSI physical designs, in Proceedings of the 48th Midwest Symposium on Circuits and Systems, Aug 2005, vol. 2, pp. 1239–1242
- [48] N. Lourenço, M. Vianello, J. Guilherme, N. Horta, LAYGEN—automatic layout generation of analog ICs from hierarchical template descriptions, in Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), June 2006, pp. 213–216
- [49] A. Unutulmaz, G. Dundar, F. Fernandez, A template router, in 20th European Conference on Circuit Theory and Design (ECCTD), Aug 2011, pp. 334–337
- [50] J. Lienig, Electromigration-aware physical design of integrated circuits, in 18th International Conference on VLSI Design, Jan 2005, pp. 77–82
- [51] J. Lienig, Introduction to electromigration-aware physical design, in Proceedings International Symposium on Physical Design (ISPD), Mar 2006, pp. 39–46

- [52] J.R. Black, Electromigration—a brief survey and some recent results. *IEEE Trans. Electron Devices* 16(4), 338–347 (1969)
- [53] T. Adler, E. Barke, Single step current driven routing of multiterminal signal nets for analog applications, in *Proceedings on Design Automation and Test in Europe (DATE)*, Mar 2000, pp. 446–450
- [54] J.-T. Yan, Z.-W. Chen, Electromigration-aware rectilinear Steiner tree construction for analog circuits, in *Proceedings of Asia Pacific Conference on Circuits and System (APCCAS)*, Nov 2008, pp. 1692–1695
- [55] J.-T. Yan, Z.-W. Chen, Obstacle-aware multiple-source rectilinear steiner tree with electromigration and IR-drop avoidance, in *Proceedings on Design Automation and Test in Europe (DATE)*, Mar 2011, pp. 1–6
- [56] X. Chen, C. Liao, S. Hu, An interconnect reliability-driven routing technique for electromigration failure avoidance. *IEEE Trans. Dependable Secure Comput.* 9(5), 770–776 (2012)
- [57] J. Jonquères, J. Portal, G. Micolau, O. Ginez, Current density aware algorithm for net generation in analog high current application, in *International conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sep 2012, pp. 153–156
- [58] I. Jiang, H.-Y. Chang, C.-L. Chang, Optimal wiring topology for electromigration avoidance considering multiple layers and obstacles, in *Proceedings of International Symposium on Physical Design (ISPD)*, Mar 2010, pp. 177–184
- [59] I. Jiang, H.-Y. Chang, C.-L. Chang, WiT: optimal wiring topology for electromigration avoidance. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* 20(4), 581–592 (2012)
- [60] Y. Tsai, T. Li, T. Chen, C. Yeh, Electromigration- and obstacle-avoiding routing tree construction, in *2013 International Symposium on VLSI Design, Automation, and Test (VLSI-DAT)*, Apr 2013, pp. 1–4

- [61] T. Adler, E. Barke, Single step current driven routing of multiterminal signal nets for analog applications, in Proceedings on Design Automation and Test in Europe (DATE), Mar 2000, pp. 446–450
- [62] P.-H. Wu, M.P.-H. Lin, T.-C. Chen, C.-F. Yeh, X. Li, T.-Y. Ho, A novel analog physical synthesis methodology integrating existent design expertise. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 34(2), 199–212 (2015)
- [63] Martins, Ricardo, Nuno Lourenço, and Nuno Horta. *Analog Integrated Circuit Design Automation*. Springer, 2017.

Chapter 3

Proposed Automation Flow

Figure 3.1 shows the flow diagram of the proposed automation tool. First, the designer would have a library containing the major topologies for analog circuits' macro cells i.e. low noise amplifiers, phase locked loops...etc. to ease the process of design. Then, after the design process, analog circuits' building blocks such as differential pairs and level shifters would be recognized. Based on the identified circuit components, a design balancing algorithm will inspect any mismatches alerting the designer and offer a suggested modifications to eliminate those violations. This step would requires the extraction of the currents and voltages throughout the circuit nodes and elements. The technology-based DRCs will be extracted too for legal placement.

Based on the identified circuit elements, an expert guided pattern generator for all circuit elements will enumerate the relevant placement patterns (i.e. common centroid and interdigitization) used in industrial standards and estimate the physical width and length of the block. After that, B*trees representing the top-level schematic will be generated and in enumerated in every possible combination. The complete set of enumerated solutions will be solved by constraint-guided linear solver in an optimal manner to yield minimum area. Optimized yielded

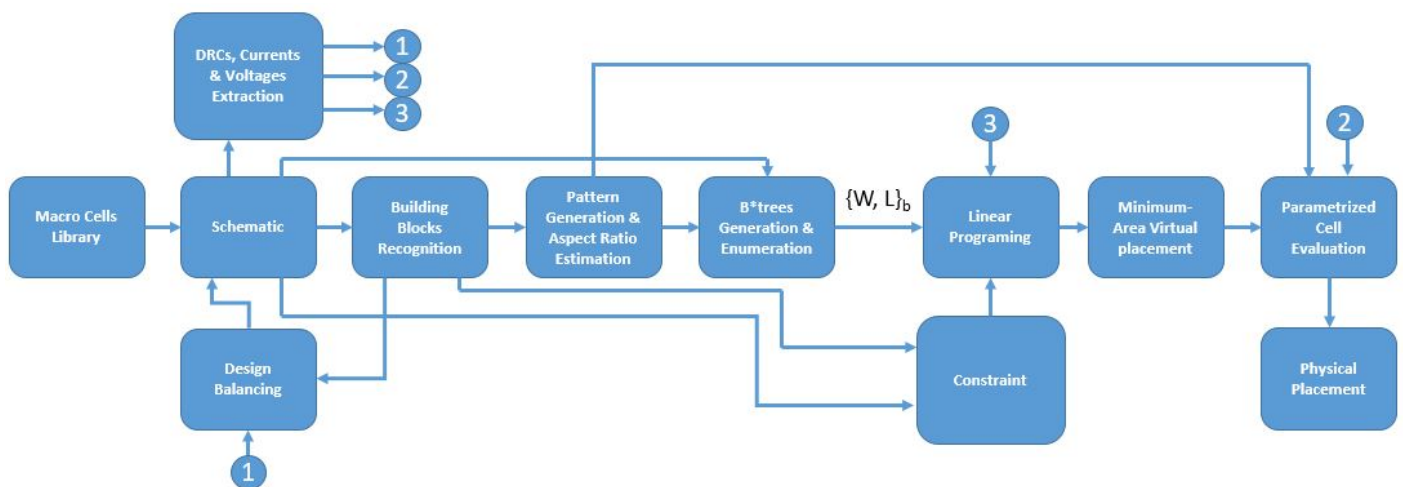


Figure 3.1: purposed analog circuit design and layout placement automation flow

solutions' virtual placement will be evaluated by parametrized cells leading to the final physical placement on the CAD tool, Synopsys Custom Designer in that case.

3.1 Building Blocks Recognition Algorithm

Analog circuits have a fundamental set of building blocks that can be integrated and tuned differently to serve various functionalities. This allows a hierarchal construction for the identification of those blocks. Figure 3.2 demonstrates this property. Based on literature review and expert opinions, the major building blocks were gathered and listed in a hierarchal manner in table 1.

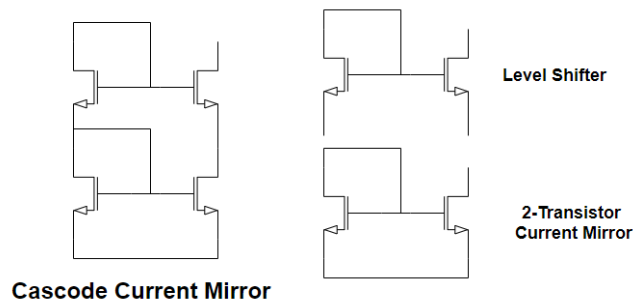













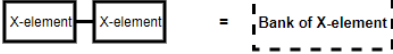


Figure 3.2: Analog circuits hierarchical construction

The algorithm starts by extracting the netlist from the schematic which contains the pin-net connections (PNC) of each transistor. Then, those connections are compared to library elements at the first hierarchal level. This has been done using connectivity matrices for level 1 library elements in order to prevent multiple identifications of the same block. The newly recognized blocks and the netlist transistors are then used to identify level two building blocks. Since the algorithm saves the constituents of level 2 blocks, only sufficient set of conditions are used instead of the exhaustive connectivity matrices used in the previous step. In order to correctly recognize level 2 blocks, the two transistors constituting level 1 elements must be differentiated. One convention could be labeling both top and left transistors "A" and bottom and right transistors "B".

Table 3.1: Analog circuits' building blocks library

Initial	
MOS Transistor	
First Level	
Diff. Pair	
Cross-Coupler	
Level Shifter	
2-Transistor Current Mirror	
Voltage Reference	
Cascode Pair	
Mirror Load	
Second Level	
Wilson Current Mirror	
Cascode Current Mirror	
4-Transistor Current Mirror	
Wide Swing Cascode Current Mirror	
Banks Special Case	
	
	

Following the results of the algorithm, cases where a transistor is identified in multiple building blocks are noticed. Figure 3.3-A shows transistor “M1” as a part of both a mirror load on

the right (A) and a level shifter of the left (B). In this case, transistors “M0” and “M1” serves the functionality of a level shifter and transistors “M1” and “M2” serves the functionality of a mirror load, hence, the multiple identification is correct. Yet, in figure 3-B, transistors “M0” and “M2” are wrongfully identified as differential pair. To avoid such cases, the whole possible combinations of building blocks are examined with one or more transistor shared. Table 2 lists the valid combinations, all invalid combinations are removed. Finally, if two equal identified building blocks are connected or share common transistors, they are added into a bank structure of the same type. Algorithm 1 shows the complete pseudocode of the building blocks recognition algorithm.

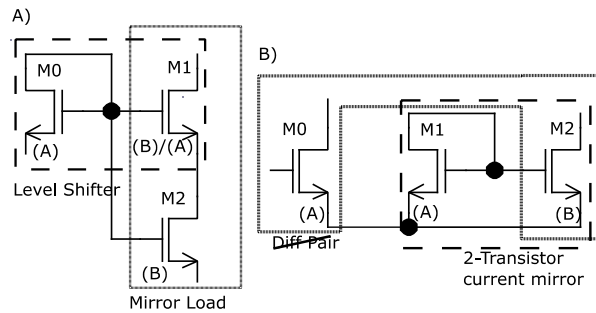


Figure 3.3: A) Correct Multiple Identification B) Wrong Multiple Identification

Table 3.2: Multiple block recognition solution

Importance Order
(Least important) Cascode pair (CP), Diff pair (DP), Level Shifter (LS), 2T-Current Mirror (2TCM), Mirror Load (ML) , Wilson Current Mirror (WCM), Voltage Reference (VR), Cross coupled (CC), 4-Transistor Current Mirror (4TCM).(Most important).
<u>Action:</u> Remove the least important
Accepted Multiple Recognitions
$\{LS^B, CML^A\}, \{CM^B, CML^A\}$

Algorithm 3.1: Building blocks recognition pseudocode

Input: PNC of n devices, connectivity matrices of each library element of the first level, the conditions of forming the second level, importance order of each library element and the accepted multiple recognition.

Output: Recognize the building blocks of the schematic.

```
1: let  $M$  be the set of each device PNC (DPNC) where each DPNC can be an ordered pair of a lower level DPNC,  $m_i = (m_j, m_k) \in M$  and  $M_i$  being the set of all DPNC in the initial level extracted from the schematic;
2: let  $L$  be the set of library elements' connectivity information,  $l_i \in L$  and  $i$  is the importance order of the library element  $l_i$ ;
3: let  $M_R$  be the set of recognized first and second level DPNC;
4: let  $\{l_i^l, l_j^k\} \in \text{Acc}$ , where  $\text{Acc}$  is the set of acceptable multiple recognitions where  $i$  is the order and  $l$  the labeling of the library's submodule  $l_i^l$ ;
5: ConInf ( $m_i, m_j$ ); Returns the connectivity information of  $m_i$  and  $m_j$ 
6: formBank( $m_i, m_j$ ); form a bank of  $m_i$  and  $m_j$ 

7: for each  $m_i$  in  $(M_i \cup M_R)$  do {
8:   for each  $m_j$  in  $(M_i \cup M_R - m_i)$  do {
9:     for each  $l_k$  do {
10:      if (ConInf ( $m_i, m_j$ ) =  $l_k$ ) then
11:         $M_R \leftarrow M_R \cup \{(m_i, m_j)_k\}; \}$  } }
12: for each  $(m_i, m_j)_k$  in  $M_R$  {
13:   for each  $(m_i, m_m)_p$  in  $(M_R - (m_i, m_j)_k)$  {
14:     if (  $((m_i = m_i) \wedge (\{l_k^A, l_p^A\} \notin \text{Acc})) \vee ((m_i = m_m) \wedge (\{l_k^A, l_p^B\} \notin \text{Acc})) \vee ((m_j = m_i) \wedge (\{l_k^B, l_p^A\} \notin \text{Acc})) \vee ((m_j = m_m) \wedge (\{l_k^B, l_p^B\} \notin \text{Acc})) \wedge (k > p)$  ) then
15:        $M_R \leftarrow M_R - \{(m_i, m_j)_p\}; \}$  } }
16: for each  $m_1 = (m_i, m_j)_k$  in  $M_R$  {
17:   for each  $m_2 = (m_i, m_m)_k$  in  $(M_R - m_1)$  {
18:     if (  $(m_j = m_k)$  )
19:       then  $M_R \leftarrow M_R \cup \text{formBank}(m_1, m_2); \}$  }
```

3.2 Pattern Generator & Physical Aspect Ratio Estimation

This subsection was adopted from the work of a team of students in Cairo University within a framework of collaboration established with them under Si-Vision IC solutions Corporation supervision to work on solving the layout automation problem.

3.2.1 Common Centroid Matching

In this approach, we try to go through the same flow as the logic followed by the engineers, instead of dealing with constraints equations. Using the expertise of industrial engineers, Quarter Cell approach has proven to be the best technique in terms of matching of large circuits.

Quarter Cell approach [Fig.3.4] means building only one quarter of the matching pattern of the whole current mirror (shaded area) then mirroring it to the other side to build a half (2), and to implement the cross-quad technique [Fig.3.5], we flip this half horizontally then vertically.



Figure 3.4: Quarter Cell: (3) is the flipped version of (2) & (4) is the flipped version of the shaded part

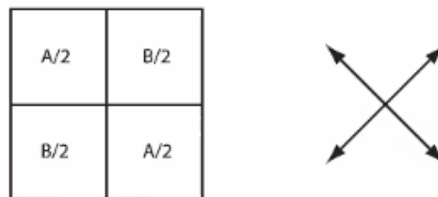


Figure 3.5: Cross Quad technique [1]

This Tool is divided into two main subsections, one before creating the pattern and the other is creating the pattern itself. In the first subsection, each device is put in one of 2 sets according to the product of the number of fingers and multipliers; either the set of fours or set of twos. Where the set of fours contains the devices, which can be put in the Quarter (replicated four times), while the set of twos contains the devices which can be put in the Half (repeated two times or their remainder after the division by 4 is 2).

Example:

Device A: 4, Device B: 6, Device C: 2

Set of fours: [A, B]; Set of twos: [B, C]

Then, we proceed to calculate the total (the sum of product of each device) and create a list of pairs. These pairs represent the number of rows and columns of the expected matching pattern. This list is ordered according to the aspect ratio and one pair will be chosen according to the floor planner's choice. In addition to the number of rows and columns the exact length and width of the pattern is provided to the floor planner taking into consideration the added space for routability. The second subsection, the devices are arranged with two main considerations: first, the diode connected device (reference) is placed in the center; second, number of devices in the same row is minimized for better routability.

3.2.2 Interdigitization Matching

Interdigitization is a very good, and simple technique. As only alter the way the components are laid out, nothing more. One can use this technique with any number of components, just need to choose how to interleave them, and how many of them to include. It can be used not only with transistors, but with any device. You can interdigitate all sorts of things, provided you have two or more of them. By this kind of matching all devices on the same row of the matching pattern experiences the same effect from the stress point of view, the temperature gradient and experience same parasitic effects.

This tool is designed to generate an interdigitated matching pattern for any 2 devices having the same parameters as width , length , number of fingers and number of multipliers , it's inputs are the parameters mentioned above , the user has the option to choose between (matching by 2 fingers) or not , then according to the information that the tool was given , it starts to generate the matching pattern for the user to decide which pattern is the best for his floorplan. Based on the decision of the source sharing took by the user, the tool will either generate a standalone finger in the matching pattern or 2 fingers matching as a single device in the matching pattern, the below placement of transistors follows our rules of matching: The devices are close to each other and they all follow the same orientation.

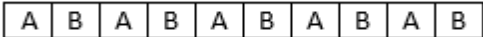


Figure 3.6: Finger by finger matching case example: two transistors no source sharing

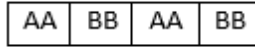


Figure 3.7: 2 fingers by 2 fingers matching case example: Two transistors matched by 2 fingers shared

3.3 B*trees generation & enumeration

3.3.1 Constraints in analog layout:

One of the pivotal aspect in analog layout is device matching. Matching is done to ensure that devices that the identicalness of each device in the design is met when the design is based on it. To ensure that the layout conforms to these stringent matching requirement, many constraints are put to meet this goal. Inherently the mask designer has these constraints in mind [2]. However, there are distinct features of these constraint that can be formalized and in the same regard these constraint can be too ambiguous to be defined. Marlot devised criteria to distinguish between them both. The formalized constraints are the constraints which is:

1. Clearly expressible in a definitive condition.
2. Only interpretable in a single way leaving no way for misinterpretation
3. Can be checked rigorously by mathematical formulae.

On the Other hand, a non-formalized constraint are:

1. Fuzzy and carry requires comprehension of the operation theory.
2. Cannot be formalized into ready check rules and is heavily opinionated.
3. Cannot be computerized.

We will focus on the formalized constraint since they can be solved algorithmically. The most common formalized constraint: is common centroid and symmetry

Common centroid constraint: common centroid is met when the centroid of the modules for each group of module is the same:

$$X_C(A) = X_C(B)$$

where X_c is the coordinate vector of each group centroid defined as follows: if x_m are the all the modules,

$$x_c = x_m + \frac{w_m}{2}$$

where x_m is the left lower x-axis position of the m module and w_m is the width of the module:

$$y_c = y_m + \frac{h_m}{2}$$

where y_m is the left lower y-axis position of the m device of the module.

$$X_m = \begin{pmatrix} x_c \\ y_c \end{pmatrix}$$

$$X_c = \frac{\sum_{module_m} w_m \cdot h_m \cdot X_m}{\sum_{module_m} w_m \cdot h_m}$$

a1	b2	b3	a4
b1	a2	a3	b4
b5	a6	a7	b8
a5	b6	b7	a8

Figure 3.8: Common Centroid Matching pattern

Device proximity constraints: in order to minimize the wafer local variation effect on the matched device, placing them closer to each other minimize the variation and enhance the matching.

Symmetry constraints: Placements should reduce the efforts in routing and allow symmetric routing to match the parasitics of the node required to match if the node parasitics is critical. As in the case of figure 3.5 the circuit functionality is degraded if the parasitics at O_p and O_n is different, symmetric routing in this cases achieve the parasitic matching, however symmetric routing has to be done on a symmetric place layout

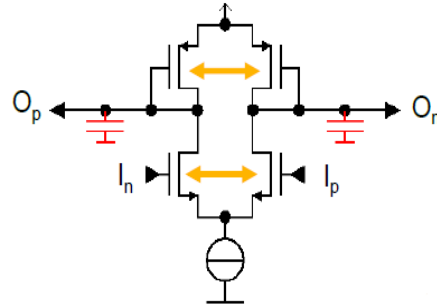


Figure 3.9 Symmetric placement

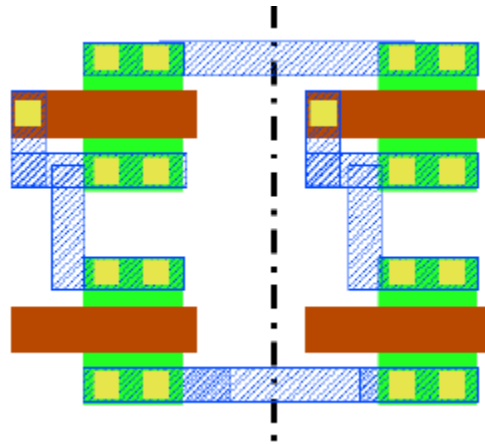


Figure 3.10 Symmetric placement and routing

3.3.2 Different representation:

B*-tree is a binary tree describing a given placement using a topological relation of each block with respect to the other. This binary tree represent a packed blocks with a packing toward the bottom left. The corresponding packed topological placement is described as the root node being the left bottom node. The child of each node is related to it topological parent in this regard, the left child is positioned above the parent node having an overlapping x-axis projection, while right child is position right to the parent block with an overlapping y-axis projection. So the B*-tree is a description of the constraint of each topological order of a given placement.

The B*-tree packing algorithm of each B* tree can be used to make a compact of placement of a given B*-tree. As in the given figure 3.7 B*-tree is used to generate the placement in figure 3.8 by the packing algorithm.

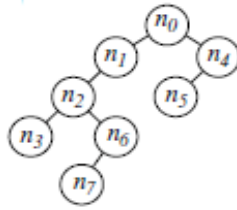


Figure 3.11: B*-tree example

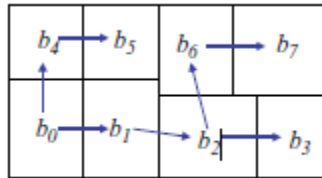


Figure 1.12: Compact placement

3.2.3 Advantages:

This topological method description of blocks is more search space efficient than the other alternative in the literature, sequence pair as Balsa [2] deduced that for more than 3 blocks the number space of different B*-tree is lower is smaller than for the sequence pair.

Obtaining the optimal B*-tree that yield the most efficient area requires complete enumeration of the possible B*-trees, the enumeration algorithm generates all the possible B*-trees. Since B*-trees structurally identical to binary trees, we propose dissecting the problem into 2 parts:

- I. Finding all the structure for a given number of nodes.
- II. Finding all the permutation of the nodes for each structure.

Different structures are found recursively since a the binary tree has a left and right child and each child can be thought of another binary tree we can find every possible children available for the tree and eventually finding all the possible binary tree structure for the given number of nodes as illustrated in algorithm 3.2

Algorithm 3.2: *B*trees generation & enumeration algorithm*

```
MakeBtree( int i) {  
  
    Btree btree;  
  
    ListBtree listBtree;  
  
    Case i: 0  
  
    listBtree.add(NULL)  
  
    return listBtree  
  
    Case i: 1  
  
    listBtree.add(btree)  
  
    return listBtree;  
  
    if (i>1){  
    for (int j=0; j<i; i++){  
        listLeftBtree.= MakeBtree(j);  
        listRightBtree.= MakeBtree(j-i);  
        for(int l=0;l< listLeftBtree.size(); l++){  
            Btree.left =MakeBtree(l);  
            listBtree.add(Btree)  
            return listBtree;  
        }  
        for(int k=0;k< listLeftBtree.size(); k++){  
            Btree.left =MakeBtree(k);  
            listBtree.add(Btree);  
            return listBtree;  
        } }  
}
```

After all the generated Btree are formed, all the Btrees are populated with a given permutation of blocks. if there are n blocks, there will be a n! Available different permutation of the blocks. So the number of a given B*-tree for a given no of nodes f(n) is:

$$f(n) = k(n) * n!$$

$$k(n) = \frac{1}{n+1} \binom{2n}{n}, \text{ where } \binom{\alpha}{n} = (\alpha(\alpha-1) \dots \frac{\alpha-n+1}{n!})$$

For n = 4, f(4) = 14*24 = 336 and consequently there are 336 different B*-tree.

3.4 Linear-Programming-Guiding Constraints Generation

To generate the symmetry constraints, Erik suggested using a directed graph tree that describe how the signal path through the circuit namely SSFG tree [Erik]. The algorithm briefly build a SSFG graph for a given schematic after recognizing the building blocks of the circuit. Each building block of the circuit is given an SSFG graph that describe its connections and functionality.

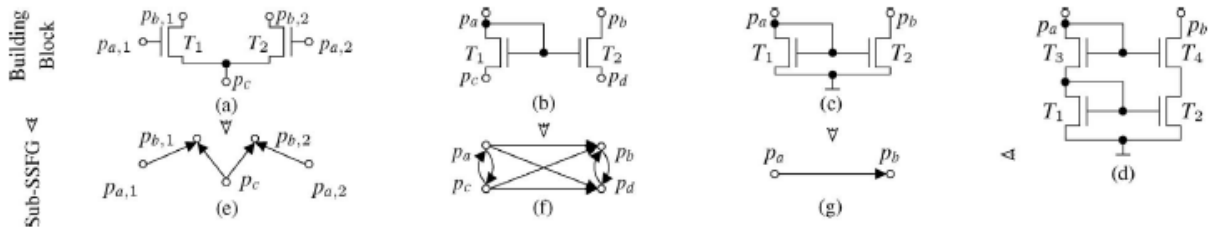


Figure 3.13: SSFG graph building

Each nodes of the circuit that is part of a basic building block is used for building the graph after each node is related to its adjacent node in the building block and algorithm link the node of each graph to nodes connected to it from the other graphs. As in the figure 3.14 the schematic is firstly analyzed for the building blocks and the building blocks' nodes are connected by a directed graph of SSFG that describes the qualitative of description of the how the signal pass through building blocks. The algorithm starts by identifying which nodes that are symmetric and then determine the symmetric blocks from these nodes. The nodes at the input of a differential pair are always assumed to be symmetric. The algorithm parse through the graph tree initially at the differential pair nodes.

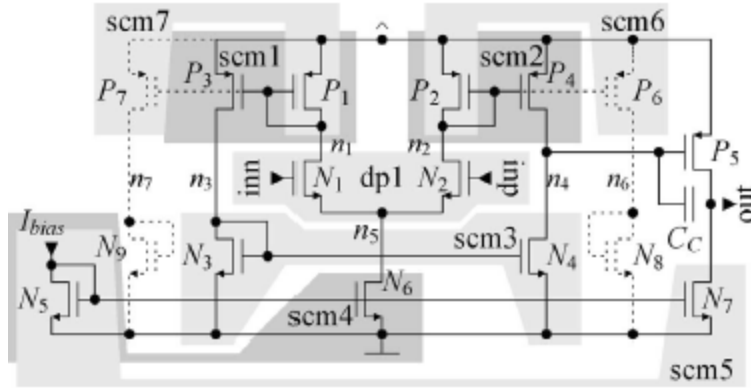


Figure 3.14: A schematic of a circuit with the building blocks recognized

The edge between each node is attributed as in figure 3.13. The algorithm search for edges of the same attributes for the left and right pair of nodes of differential pair the edge is assumed to be symmetric and also for any symmetric node the algorithm search for the edge starting from them that are equal in attributes if they are equal they are recognized as symmetric. If the algorithm do find edge of equal attributes it searches for an edge that connect the 2 search nodes together and stops and this step. Each node that the symmetric edge point to is symmetric. After the symmetric nodes are identified the build blocks connected to these nodes are recognized as symmetric.

3.5 Linear Programming

Each B*-tree can be used to generate horizontal and vertical constraints. These constraints are met only by blocks that follow the same exact B*-tree's topological relations. The algorithm for generating the B*-tree to Vertical Constraint graph and Horizontal constraint graph is presented in [3].

Algorithm 3.3: *B*trees to VCG*

```

buildVerticalCG(CGNode thisNode, predecessor)
if B*-tree node of thisNode has a left child then

```

```

    leftNode← new CG node for left child;
    add edge from thisNode to leftNode;
    buildVerticalCG(leftNode,thisNode);
else
    add edge from thisNode to the end node;
    if B*-tree node of thisNode has a right child then
        rightNode← new CG node for right child;
        add edge from predecessor to rightNode;
        buildVerticalCG(rightNode,predecessor);

```

Algorithm 3.4: *B*-trees to HCG*

```

startNode← new CG node as start;
endNode← new CG node as end;
create list of y regions of all modules;
initialize all y regions with startNode;
forall modules in preorder do
    modNode new CG node for module;
    forall regions r in region list from module.y to module.y module.height
    do
        shadowNode← node of the module which is registered in r;
        add edge to modNode from shadowNode;
        register module in r; // Shadow that segment
    remove multiple edges;
add edges from all nodes in region list to endNode;

```

As in figure 3.15, the algorithm, B*-tree to VCG is used to build VCG from B*-tree, initially the A is linked to the start and since B is a left child it is above A, therefore linked to it in A in a similar manner D is linked to C. While B is a right child meaning it is on the same level of parent or linked to Start in the VCG.

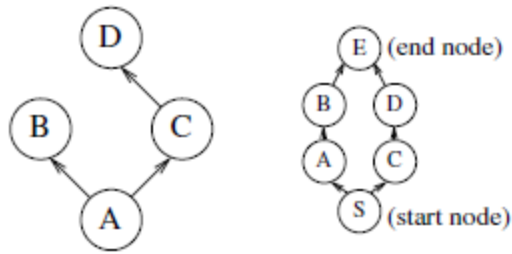


Figure 3.15: B*-tree to VCG

In figure 3.16 the HCG generation require the knowledge of the height of each block, and build the HCG by parsing the B*-tree in a preorder manner, the preorder of the B*-tree, reach the right child after the parent therefore the order of reach describes horizontal constraints, each block that has an overlapping y-axis projection with a placed block is linked to in the HCG.

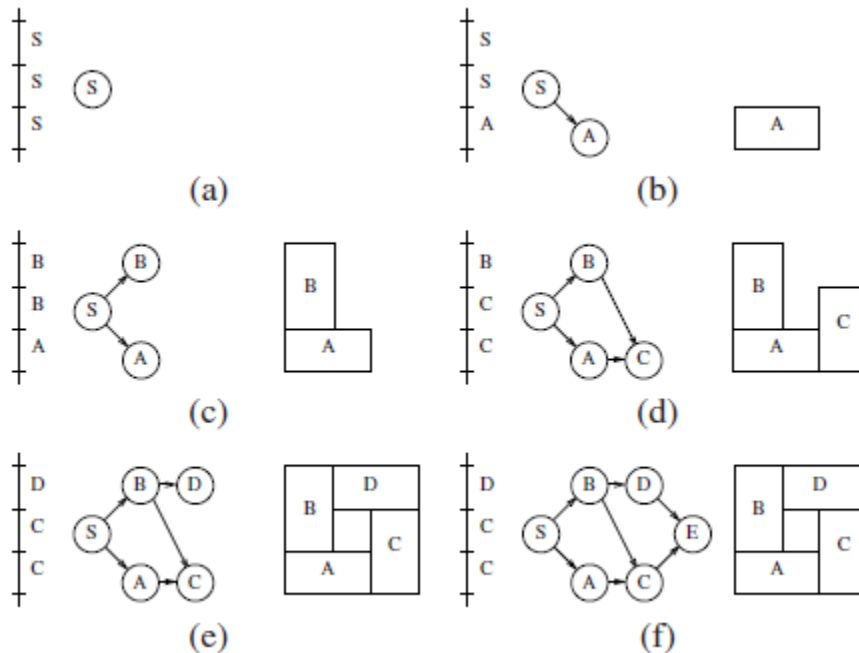


Figure 3.16: B*-tree to HCG

All these constraints in addition to other constraints are solved using Microsoft Z3, Z3 requires a certain language SMT-LIB2, starting from the constraints and the dimension of

each block we can generate a code that is readily solved by Z3 to get the left most x-coordinate and the bottom y-coordinate of each block.

3.6 Physical Placement by Means of Parametric Cells Evaluation

3.6.1 Current Mirror

The following flow chart is made to illustrate the flow of automating the current mirror taking into consideration the above factors that affects the design performance.

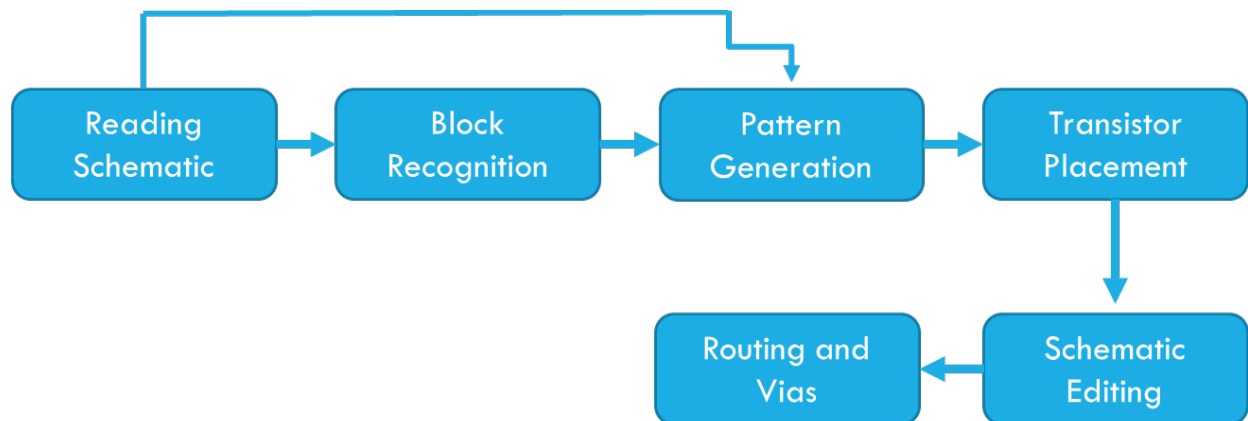


Figure 3.17: Current Mirror Automation Flow Chart

This flow chart shows that to build a current mirror, first, will start by reading the schematic. In Synopsys tool, a netlist of the schematic can be generated as extension of CDL file. This extension gives all the information about width length and every terminal connection of used devices. As illustrated in the previous chapter when we talked about block recognition, these information from the schematic will be passed to C++ code to investigate the connections and determine the different block. Hence, the block recognition code will determine what transistors are forming the current mirror. Current mirror transistors share the same source and gate.

After that comes the pattern generation step. In this step it is important to place the diode connected in the middle as it is the most sensitive device in the current mirror. Now a concept of

quarter cell is used. For a block having four quarters, the quarter in the positive x and y and the other one in negative x and y needs to be symmetric about the origin. And in the same manner the other two blocks to create a common centroid as shown in the following diagram.

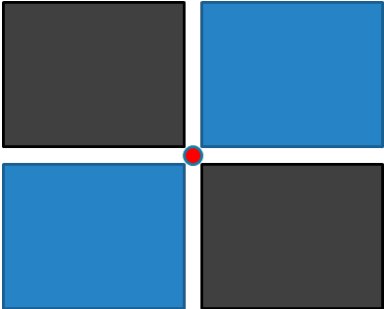


Figure 3.18: Common centroid by means of quarter blocks

Now there is some ambiguities that needs to be considered. If the pattern has odd number of rows, we cannot create a quarter block. To overcome this issue, the row in the middle is removed. Then the quarter cell concept is applied to match the devices. Then after that, the middle row is created, and it should be symmetric around the y-axis as shown in the following diagram.

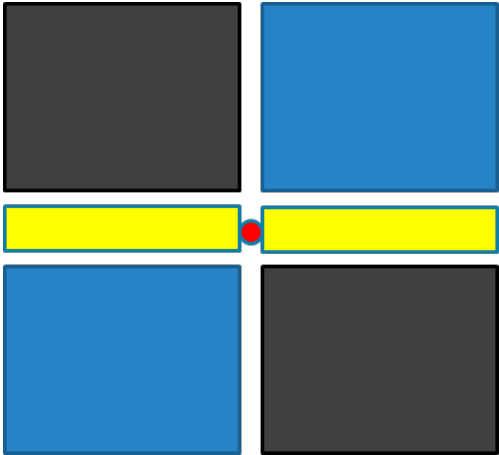


Figure 3.19: Common centroid by means of quarter blocks in case of odd number of rows

Now an important note needs to be illustrated. The pattern generation code needs to produce several alternatives that will be used later in the enhanced shape function to get the best overall floor plan. Hence, the codes will manipulate the number of multipliers and accordingly the widths of each transistor these new properties should be passed to the transistor placement block.

The pattern generation block was adopted from the work of a team of students in Cairo University within a framework of collaboration established with them to work on solving the layout automation problem.

Now, after this has been completed, it is time to place this pattern in an actual layout. A TCL language is used with combination with Synopsys Custom Compiler to place this pattern. This step is divided into two main parts which are pattern manipulation and device instantiation. The following pseudo code is made to illustrate the algorithm.

Algorithm 3.5, part a: *Pattern manipulation*

- from the giving pattern, extracting number of rows and columns.
- now by means of rows and columns we will create another array holds transistors names.
- by means of transistor names we can loop over the pattern and extract the number of multipliers for each device.
- in the same last step, we could simultaneously rename different multipliers by the device name followed by the .multiplier number.
- Finally, we have modified pattern array.

The last step is very critical. It is important to enable SDL, schematic driven language, to enable the program for checking the nets and devices and compare it with the schematic. Now the second step of the code.

Algorithm 3.5, part b: Device Instantiation

- for comparison between schematic and layout for checking any miss match, the width and length strings is manipulated. They are converted to double then a u letter is added to them.
- extract the cell name and library name from the layout as it shares the same information with the schematic.
- set the desired origin.
- check if the source and drain is shared or not. This is an input information. This will differ in the horizontal spacing.
- based on the extracted DRCs a horizontal and vertical spaces should be left.
- the vertical spaces should account for the gate contacts and substrate contact to avoid latch up.
- instantiate the devices using names from the modified pattern taking into consideration all the above requirements.

Now, comes the schematic editing. We simply read the schematic and compare its information from the output of pattern recognition block and edit the schematic based on this

information. The instantiation output is shown in the following figure. We can observe that the different multipliers is identified by the software.



Figure 3.20: Device instantiation

Finally, comes the routing step. The rout width should account the effect of electromigration. Electromigration is caused by the molecular displacement of atoms by flow of electrons over time. The metal line will break creating an open circuit as illustrated in the following figure.

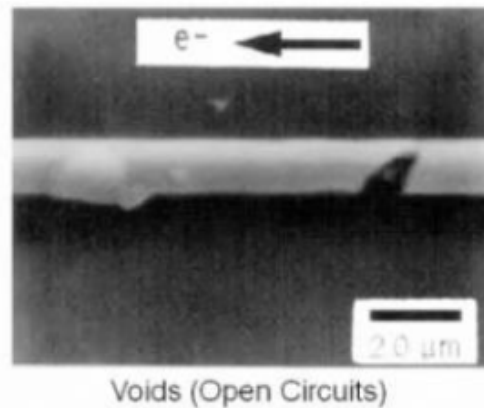


Figure 3.21: Electromigration

This is based on the amount of current the current mirror will deliver. And hence we adjust the width of the metals. Number of wires is calculated as follows. A wire to collect sources and then a perpendicular wire to collect these wires. For the drain, it is depending on the number of devices in a column. The maximum number would determine the needed vertical wires above every device. Then depending on the number of devices, horizontal wires in needed to collect the drains. Now, the total current through the drain and source is known from the circuit which will flow through the wires that collects the drain and sources. Hence, these collecting wires should has the maximum widths. Then the current will be divided through the other drain wires for every device.

3.6.2 Differential Pair

The differential pair pcell is placed and routed to achieve low parasitic mismatch in the input and output, the input and output node are the gate and drain respectively. The placement is done through an interdigitized placement algorithm with 2 rows to achieve more matching between the devices as shown in figure 16 and to isolate the noise of the substrate a guard ring is placed around it, this guard ring is automated to be at the same distance from the diffusion distance of the transistors as the tie placed in between the 2 rows to have the same stress effect from up and down on the transistor consequently allowing more matching.

The signal at the input node has to see the same parasitic as its other differential pair signal, the signal, the input routes symmetrically face the same parasitic capacitance from the tie and the transistor as the signal pass between the two transistors rows.

Similarly the routes of the differential pair at the output are routed in a symmetric way to overcome any change in the parasitics of the 2 output differential nodes.

The routing of the source is done over the transistor which is tolerable since it does not carry an oscillating signal as the source is connected to the ground.

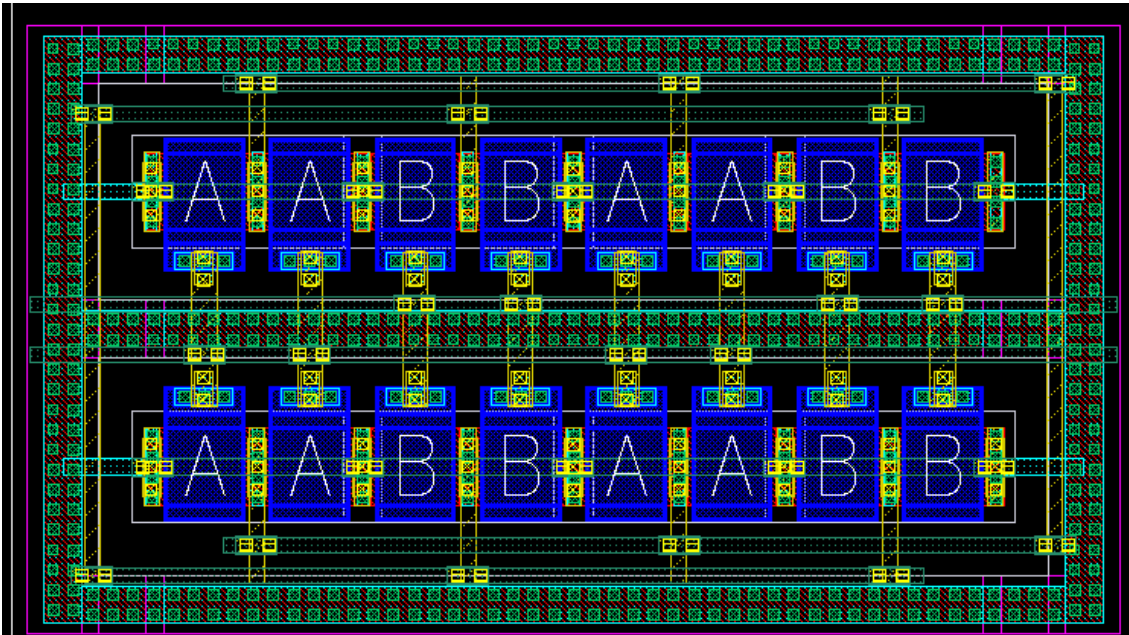


Figure 3.22: Interdigitized diff pair

3.7 Conclusion

A complete automation flow for analog layout problem was presented. Important implements within the flow was properly discussed. The next chapter discuss the results while chapter 5 will shed the light over the limitations and future work.

References

- [1] Di Long, Xianlong Hong, Sheqin Dong, “Optimal Two-Dimension Common Centroid Layout Generation for MOS Transistors Unit-Circuit”
- [2] Balasa, F., Maruvada, S. C., & Krishnamoorthy, K. (2004). On the exploration of the solution space in analog placement with symmetry constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(2), 177-191.
- [3] Strasser, M., Eick, M., Gräß, H., Schlichtmann, U., & Johannes, F. M. (2008, November). Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design* (pp. 306-313). IEEE Press.

Chapter 4

Results & Discussion

This chapter illustrates the results and analysis of each part of the flow and how they are linked together by taking a circuit schematic that employs symmetric constraints and matching constraints. The schematic is illustrated in figure 4.1.

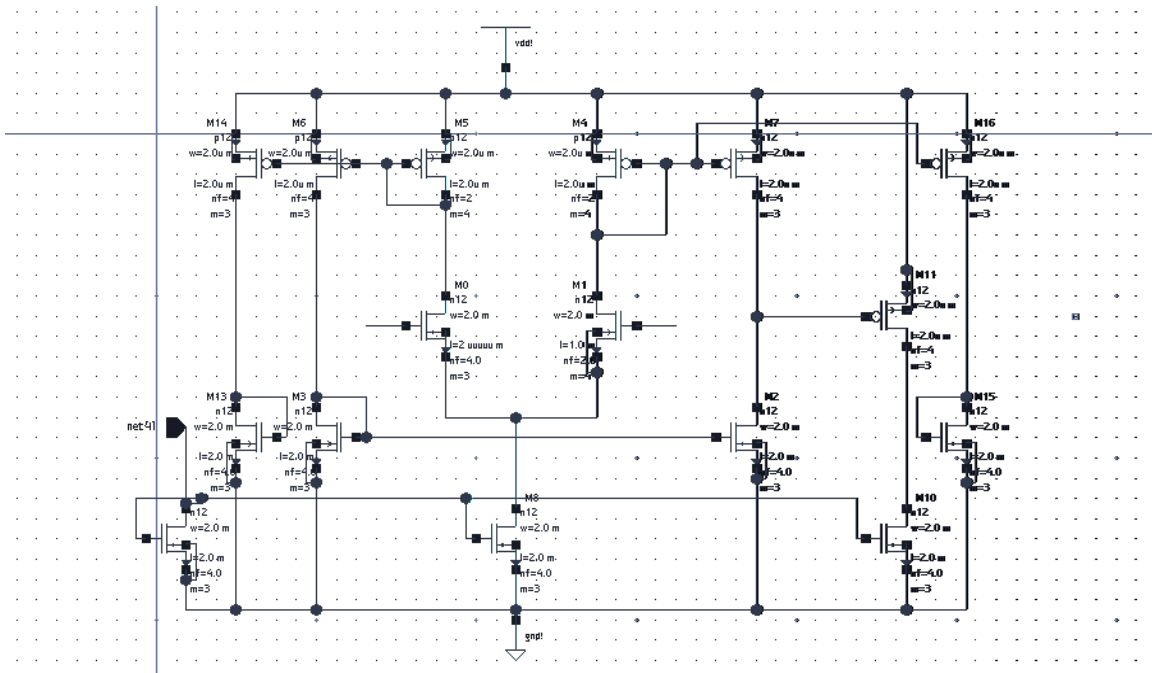


Figure 4.1 Analog circuit schematic requiring different type of constraints

This chapter is organized as follows, the first section will illustrate the recognition of the basic building blocks by employing our netlist reading C++ program following the recognition of basic blocks, after that the program will identify the required constraints for the placement of these basic blocks.

The second section will discuss briefly the pattern generated for each basic block, estimation of its area, and determining many variants of each block. The third section will illustrate the results of formalization of the constraints and solving them using SMT Z3 and outputting

different aspect ratios of the design of the final placements. The fourth and last section will illustrate the evaluated p-cell and the final detailed placement and routing of each block and placement for a selected virtual placement.

4.1 Recognizing and constraining basic blocks

The circuit figure 4.1 is translated into netlist of the format CDL using Synopsys Custom Compiler, this format readily contain each transistor number of fingers, multiplier, width and connections of each transistor. The program populate a data structure that contain the transistor's parameters and then recursively populate the transistors in a suitable basic analog block. The result of running the program is illustrated in given window of figures 4.2 and 4.3.

File	Edit	Format	View	Help
MM3	MM2	CM		
MM1	MM0	DP		
MM9	MM10	MM8	CMB	
MM5	MM14	MM6	CMB	
MM4	MM16	MM7	CMB	

Figure 4.2: Result of recognized

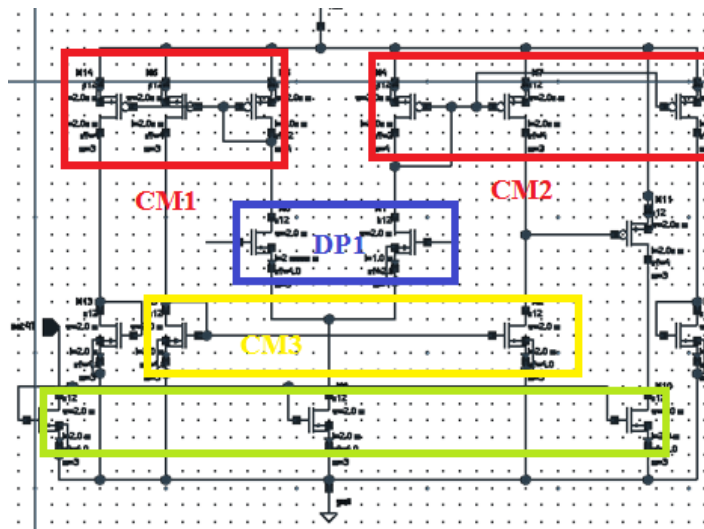


Figure 4.3: Basic blocks identified

These basic blocks are analysed using the symmetric constraints program which starts from the differential pair and determine the symmetric nets and from the symmetric nets the symmetric blocks are identified. Figure 4.4 show the symmetric nets.

```
MM5 MM14 MM6 CMB
MM4 MM16 MM7 CMB
Start
net4
net54
net55
Time taken: 0.06s
Process returned 0 (0x0) execution time : 0.090 s
Press any key to continue.
```

Figure 4.4: Symmetric nets identified

These blocks are then passed to enumeration algorithm where all possible B*-tree are generated with the non-symmetric feasible being filtered, before filtering the B*-tree there were 5040 candidate B*-tree after filtering the non-symmetric feasible for the given symmetric blocks the no of candidate B*-tree decrease significantly reaching 502. This step decreased the runtime significantly.

4.2 Basic blocks pattern generation and dimension estimation

The no of multiplier and finger of each transistor in the basic building block are used by the pattern generator designed by students in Cairo University in collaboration with Si-Vision to suggest many patterns that are common centroid for the current mirror blocks and inter-digitized for the diff pair. These patten generators are expert knowledge driven, that they take into account thee accumulated knowledge of the engineers in the industry.

These variants allow considering the blocks as a module where they decrease the space of solution significantly and also increases the quality of the resulting solutions instead of trying the different variation of the transistors in the block. The pattern is a letters that describes where the

transistor of is located in the block so AABBA is common centroid pattern describing a transistor A abutted next to another transistor with 2 B transistor abutted to each next to another 2 A transistor abutted.

4.3 Formalization of constraints and solving them

The resulting graph from the previous steps describes the horizontal relation between the basic building blocks and similarly the vertical ones, however there is a more important relation between the basic building blocks which is the symmetric constraints, the symmetric constraints are understood in this regard: the symmetric pairs are aligned with Y-axis projection being the same.

Differ pair that share the same axis of symmetry have a common geometric center. This constraints are translated into an SMT solver language, namely SMT-lib2 using code generator that parse through the VCGs and HCG and generate the equivalent lines of code as in the following code:

Algorithm 4.1: Translated conditions by the SMT solver

```
(set-option :pp.decimal true)
```

```
(set-option :opt.priority pareto)
```

```
(declare-fun x0() Real)
```

```
(declare-fun x1() Real)
```

```
(declare-fun x2() Real)
```

```
(declare-fun x3() Real)
```

```
(declare-fun x4() Real)
```

```
(declare-fun xh() Real)
```

```
(assert (= x0 0.0))
```

```
(assert (>= x4(+ x0 0)))
```

```
(assert (>= x3(+ x4 200)))
```

```
(assert (>= xh(+ x3 200)))
```

```
(assert (>= x2(+ x0 0)))
```

```

(assert (>= xh(+ x2 100)))
(assert (>= x1(+ x0 0)))
(assert (>= xh(+ x1 200)))
(minimize xh)
(check-sat)
(get-model)

```

SMT-Z3 has wide range theories that are used in proving theorems however also it can be used for optimization as in the case of our constraints each horizontal constraints is translated as an inequality where bounding box x-axis is xh and the program goal was set minimize it in order to insure having an area-optimal layout. Different constraints are run with one for each the vertical and horizontal projection of the modules. Each B*-tree of each variant of modules is run through the constraints formulizer.

After the coordinates are evaluated for each block, the block are visualized using a C++ library for game graphics namely Simple Fast Media Library, SFML, the user can browse through the available solution or can choose the lowest area of the available variations, for our example, circuit we found this variation to be most space efficient as in the figure below.

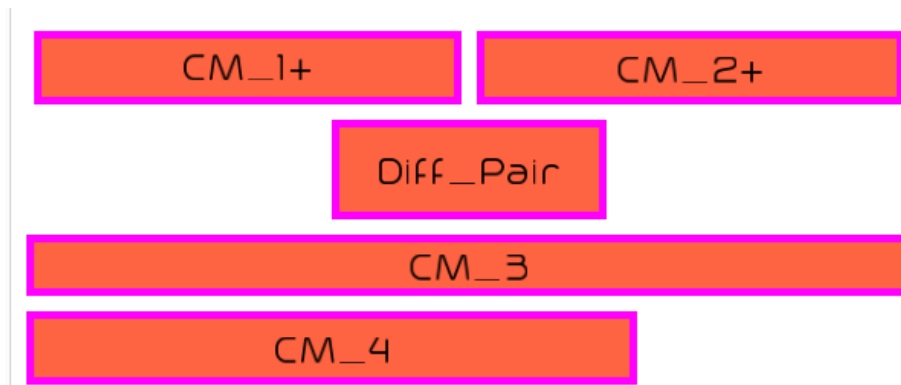


Figure 4.5: Virtual placement of the basic analog blocks

4.4 Detailed placement and routing

The virtual placement coordinates is used in custom compiler to generate the layout of each analog basic block, the analog basic block is built through a scripting language within custom compiler to automate the placement, and routing. Currently the p-cell supports each the current mirror, differential pairs other blocks are future work. Scripts of the placement and routing requires

the knowledge of the pattern and consequently require the input of both the virtual placer output coordinates and the pattern generator to place each device in its respected position. The result of the circuit is shown in figure 4.6. The analog router of Synopsys can be setup to rout the top level of the blocks but future work can work on automating the required constraints for the router to respect detailed routing.

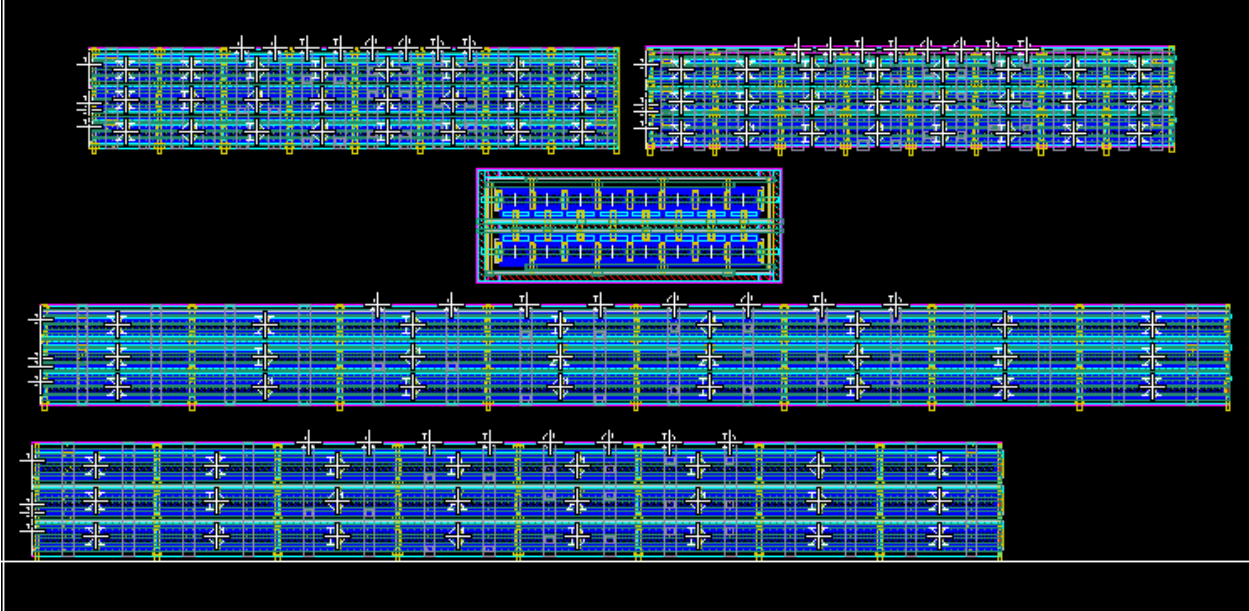


Figure 4.6: Layout of the whole circuit

The layout of current mirror as shown in figure 4.7 is routes of each row is connected vertically through metal 4 and the Horizontal routes itself is metal 3 where these routes are connected to one respected device through a Via from metal 2 and metal 3, since all the source and drain are raised to metal 2. To illustrate more clearly the details of the routing Figure 4.8 it is clear the vertical routing is wider than the horizontal since it carrier more current due to the fact that it carrier half of the current of the respected device.

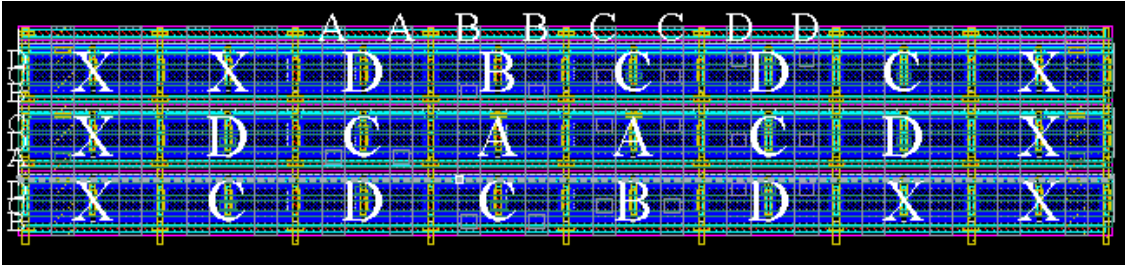


Figure 4.7: Current mirror pcell evaluated

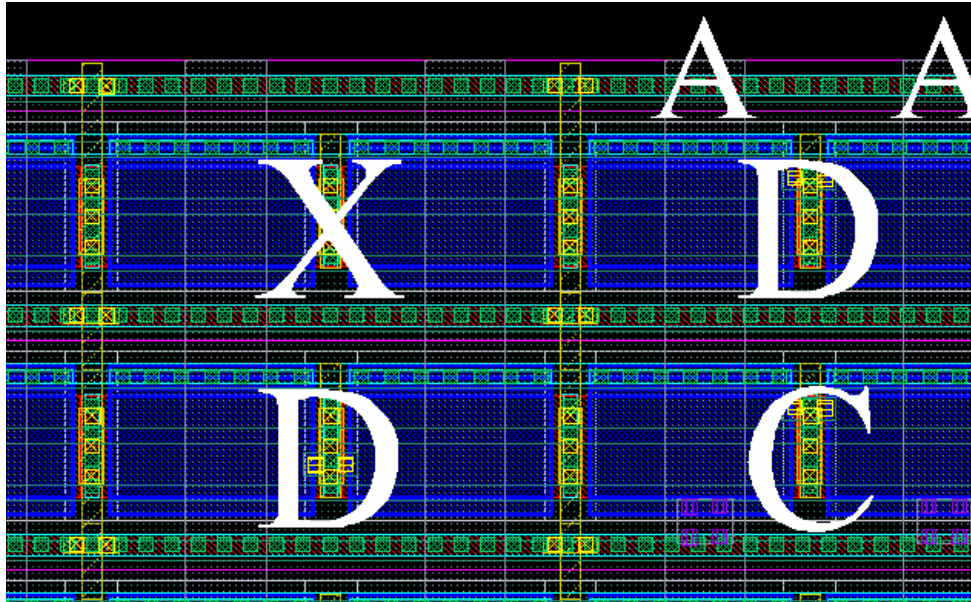


Figure 4.8: The details of the routing

More matching constraints has to be taken into consideration while designing the p-cell of differential pair as illustrated in figure 4.9 than the current mirror, since the differential pair carries a varying signals as an input at the gate and as an output of the drain. These signal can cause unnecessary coup between the input and output, so their routing are separated as far as possible and differential routing is employed to suppress any different in parasitics at each of the differential nodes.

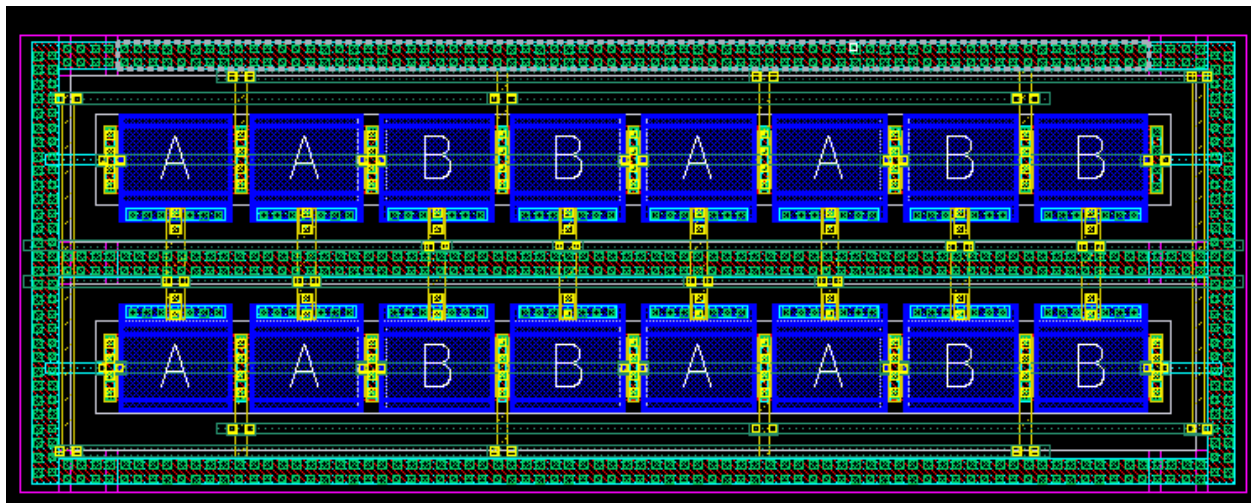


Figure 4.9: The pcell of the differential pair

4.5 Conclusion

While the flow is still not complete i.e. elements like circuit sizing. It has been thoroughly demonstrated that this flow may offer a very promising results in the future.

Chapter 5

Conclusion and Future Work

Analog layout automation is an important step as it has an impact on the analog design. Analog layout automation would decrease the headache from manual layout and decrease the effort of finding the optimum design. The process flow of layout generation and verification as follows. First, we have the schematic and constraints. Then, floor planning, routing, physical verification, PEX, and finally post layout simulation to ensure a working design after the added parasitic from the generated layout. Several layout generators is made throughout history. For example, ILAC, ALSYN, LAYLA, ALG, AIDA-L, etc.

For analog topologies constraints that needs to be put in consideration are device matching, symmetry and proximity. Symmetric devices as differential pair needs to be placed and routed symmetrically. Matching also needed in several blocks to eliminate relative errors between devices parameters. Moreover, proximity is critical for devices that communicates together a lot. After that floor planning needs to be made. It is either done deterministically or stochastically. Here our work focus will be on deterministic approach specially Plantage. An example of a stochastic algorithms is simulated annealing.

To be able of handling different blocks, we need to define means of representing the blocks. We could use either the center of each block or the down left corner be consistent. Hence, we define x and y coordinate for each block. Another method of representing the blocks is using trees. Moreover, trees is used to represent the alternative positions between different blocks. Some examples of used trees are binary trees, B* tree, and HB* tree.

Plantage is a fully deterministic placement. It is based on hierarchal structure of analog circuits. The hierarchy is based on the building blocks and the constraints generated by them. Plantage uses enhanced shape function to be able of adding different blocks together to obtain the best overall or global placement. Some factors to consider in placement is reviewed as thermal driven placement, current driven placement, stochastic optimizer.

Our flow is as follows, first, having a macro cells library containing the major topologies for analog circuits as LNA, PLL for design ease. After that, variety of information is extracted from the schematic including devices connections, width, length, exc. Also, from library files we extract the DRCs and parasitic information. After that, the building analog blocks is recognized based on the device connections. Now, detailed placement is made for the building blocks. Here, we tackled current mirrors and differential pairs. The next step was to add these building blocks together by means of enhanced shape function or linear programming. We compare the produced overall layout to get the minimum area virtual placement.

Building block recognition is based on the device connections. The extracted device connections are compared to library of known elements we call it first hierarchical level 1. This illustrates that the library is divided to several levels starting from the transistor and making connection between blocks in the higher levels. Hence, we are able to identify the differential pair and current mirrors as well as other blocks.

Pattern generation is the next step after recognition. For current mirror, a common centroid matching type is made, while, for diff pair interdigitation matching method is made. However, for both matching a pattern needs to be generated. A variety of patterns is generated such that we have different solutions to iterate over in the global placement in with the enhanced shape function. The common centroid pattern is based in quarter cell. The algorithm ensures that the opposite corners through the origin are identical.

The linear programming or enhanced shape function uses B*tree to generate horizontal and vertical constraints graphs. Then comes the global placement that should account the symmetry. Now the routing should also satisfy the symmetry constraints and account for voltage drop to not exceed a maximum value by adjusting the wire width to alternate its resistance.

Finally, the result was shown for an example. The flow is processed over the given circuit design to show the output of each block.

Appendix

A1: Concept of Analog Layout

Analog layout is a very critical process due to the sensitivity of device parameters on the performance of analog blocks. Hence, layout techniques are developed to reduce these effects. In the beginning, moors law suggested that the transistor number on chip would duplicate every 18 months. This leads to complexity would increase as the size of devices would decrease. Smaller size means larger relative error as shown in the figure below.

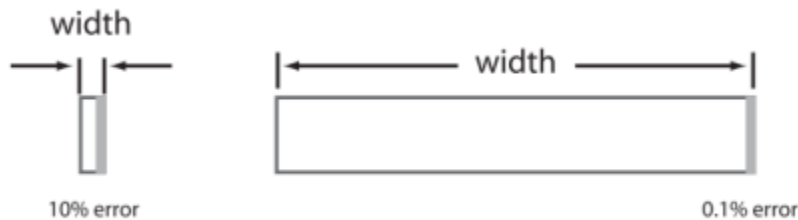


Figure A1: Size effect on error

Hence, this shows how it is becoming harder to produce the desired parameters. Therefore, the layout design should ensure working design even with all these process variations. These process variations introduced from several factors as mask alignment. The first step in the photolithography process is to develop a mask which is made using chromium pattern on a glass plate. Then the wafer is coated with photoresist, which is a polymer, using spinner coating adjust its speed to ensure a photoresist uniform thickness with a specific selectivity. After that the photoresist is subjected to ultra violet light. If positive photoresist is used, then the areas subjected to light will be softened and then removed by a developer. On the other hand, if negative

photoresist is used the areas covered by the chromium which is not subjected to light will be softened and removed by the developer. Hence, mask alignment is critical step in fabrication as it indicates what will be removed. There are some errors might be produced from mask aligning. Moreover, mask could produce more errors aside from aligning. For instance, there is printability issues in lithography as the wave length is greater than the device dimension which will cause the lithography pattern to be faded away and not as desired as illustrated in the following figure.

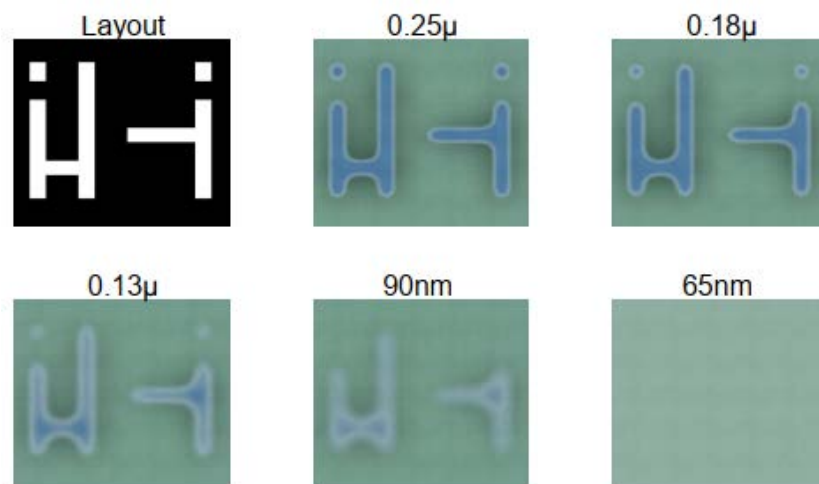


Figure A2: Printability Issues in Lithography

To overcome this issue a manipulation technique in the structure shape called optical proximity correction (OPC) is developed. This technique adds small objects to photomask areas that frequently is unprinted as enlarging corners and adding bias lines to the edge of features as illustrated in the following picture.

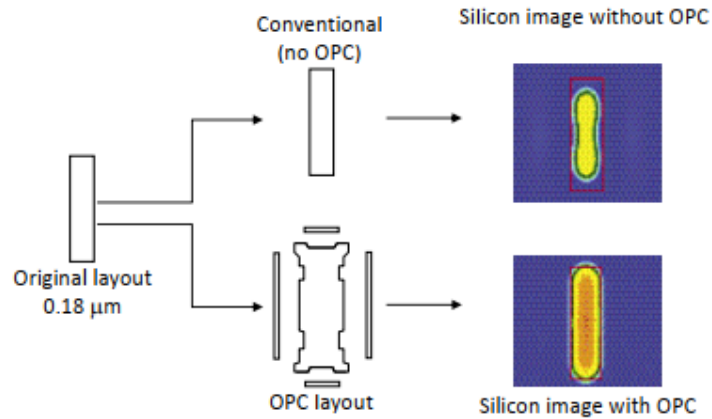


Figure A3: Optical proximity correction

Moreover, another technique called coloring is used to print the closed structures without becoming overlapping. It is a double patterning technology (DPT) which splits single layer into two separate masks. Hence, it requires two color layout decomposition process to indicate which features will be placed on which mask. The following picture further illustrate this technique.

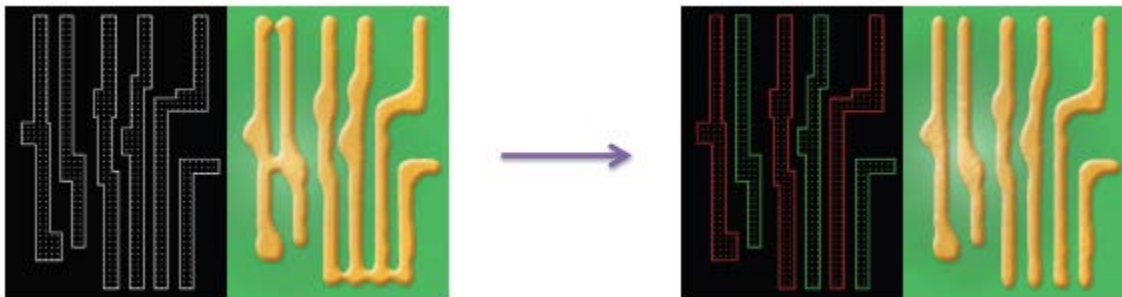


Figure A4: Double patterning technology

Furthermore, problems in manufactured device might come from the etching process. After masking we etch the oxide and then strip the photoresist. After that, we can diffuse n or p, implant or grow a different quality oxide on the wafer. In this step, a phenomenon known as well proximity effect (WPE) occurs. Simply, the transistors that are closed to the well edge are subjected to more

dopants because of scattering from the well edges. The increased number of dopants will cause the performance of these transistors to deviate. V_{th} is proportional to the square root of doping as illustrated in the following equations.

$$V_{TN} = \frac{|Q'_{SD}(\max)| - Q'_{SS}}{C_{ox}} + \phi_{ms} + 2 \phi_{fp}$$

$$\phi_{fp} = V_{thermal} \ln\left(\frac{N_a}{n_i}\right)$$

$$Q'_{SD} = e N_a x_{dT}$$

$$x_{dT} = \sqrt{\frac{4 \epsilon_s \phi_{fn}}{e N_d}}$$

$$Q'_{SD} \propto \sqrt{N_d}$$

The doping is N_d , hence, Q'_{SD} is proportional to square root N_d and because \ln function is weaker than the square root function, we can argue that the threshold voltage is proportional to the square root of the doping. Therefore, well proximity effect is very critical as it affects the threshold voltage of the devices near the edge well. As a result, it is important to add dummy transistors to keep the important devices that is responsible of the function as near as the desired value. Later we will handle how to add dummy devices as it is a critical step in some sensitive devices as differential pair. Moreover, this change in the threshold voltage can vary the transistor speed by $\pm 10\%$. Speed is represented in terms of current flow through the transistor. If the threshold voltage increased the speed will decrease as

$$I_D \propto (V_{GS} - V_{th})^2$$

Well proximity effect also adds stress effect on these transistors as the on region for the boundary transistor is less than the rest of transistors as they have higher V_{th} .

Furthermore, another process that also has impact on the device performance is oxidation. Oxidation is a process which converts silicon on the wafer into silicon dioxide. The chemical reaction between silicon and oxygen starts at room temperature and then stops after a very thin oxide film. Because the oxygen atom is larger than the silicon, it adds stress to the nearer areas. Hence, beside trench isolation the device will encounter high stress. Hence, it is important to consider these effects in the layout and know its effects on the device parameters to ensure working design.

Therefore, Matching is very critical step in analog layout. Matching purpose is to make every device sees the same variations in process, voltage, and temperature (PVT). Hence, no relative error between them. In differential pair for example each device must exactly have the same parameters for them to functioning correctly. In matching a unit is chosen for appropriate matching. For instance, matching the bellow resistors,

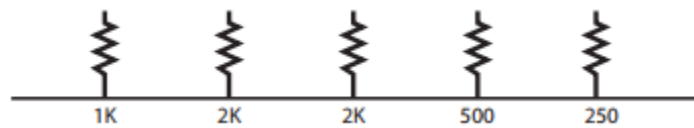


Figure A5: Devices to be matched

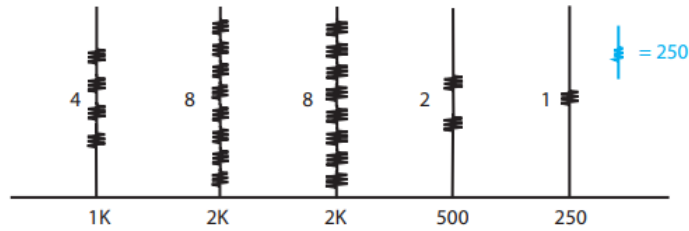


Figure A6: Choosing matching element of 250 ohm

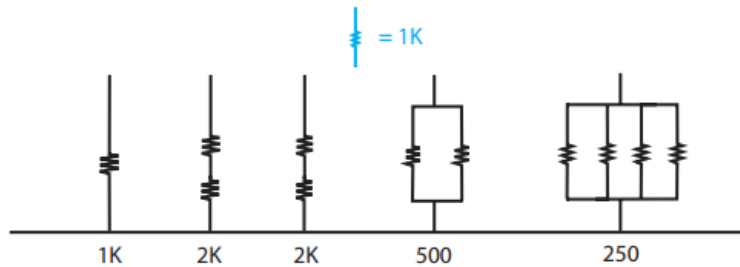


Figure A7: Choosing matching element of 1 kΩ

Hence, as a conclusion from the figures we find that it is more appropriate to choose a matching element of 1Kohm as it produces less elements to be matched.

There are two common ways for performing matching. The first is called common centroid. Simply, all the devices will share the same geometrical centroid as illustrated in the following picture.



Figure A8: Common centroid

For linear stress profile, using common centroid will insure that all the devices encounter the same stress level.

The other matching method is called inter-digitization. It is illustrated in the following figure.



Figure A9: Interdigitization

Simply, place alternate components on after the other. For nonlinear variations as non-linear stress, this method will approximate the overall net stress to be the same as every device is placed in the approximately the same region. In differential pair, it is most common to use inter-digitization method so that the two devices seek to have the same effects. Therefore, minimizing even the non-linear effects. On the other hand, for current mirror, it is more common to use common centroid and approximate the stress in linear form.

Finally, some effects that needs to be keep in consideration in making the building block are latch up and substrate noise. Latch up is a generation of low impedance path between the supply and the ground from the interaction with PNP and NPN bipolar parasitic transistors as illustrated in the following figure.

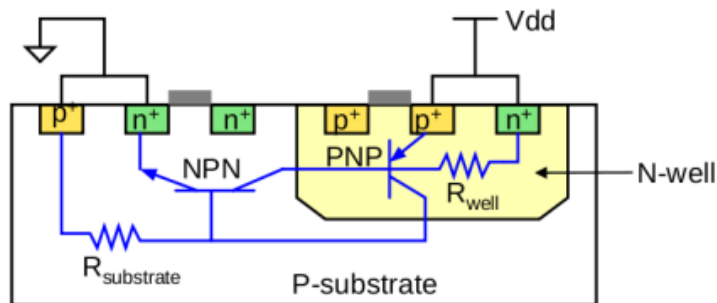


Figure A10: Latch Up

If R_{well} or $R_{\text{substrate}}$ increase it will trigger this effect. If a spike happened on Vdd or GND it will make a voltage drop on these resistances and make the reversed biased junction to be forward and hence a path is created. This problem is detected in the DRCs, and to solve it we need to add substrate contacts and bias it to Vdd or GND depending on if it is in the PMOS or NMOS. Doing so, we are decreasing R_{well} and $R_{\text{substrate}}$. The other factor to consider is substrate noise. The switching activities of the digital block generates noise that is biased to the substrate. These noises can navigate in the substrate and reach the analog blocks. Hence, to avoid this from happening we need to isolate these blocks using guard ring and bias it so noise will go to GND or VDD and stop navigating the substrate.

This section was loosely based on: Christopher Saint, Judy Saint-IC Mask Design_ Essential Layout Techniques-McGraw-Hill Professional (2002)

A2: Classification of floorplan representations for analog design automation

Adopted from the book: Analog Integrated Circuit Design Automation: Placement, Routing and Parasitic Extraction Techniques

Table A1 Classification of floorplan representations for analog design automation

Representation	Advantages (+) Drawbacks (-)	Symmetry ^a	Perturbation complexity ^b	Packing complexity ^b
Absolute [4, 6-8, 10-12]	(+) Every possible placement can be described; (+) Optimal constraint handling, i.e., no packing, structural scan or post-processing time is required; (-) Allow illegal overlaps during moves; (-) Slower due to the solution space infinitely large.	✓	$O(1)$	$O(1)$
Relative	(+) Smaller solution space than absolute; (+) Moves are modifications in the relative positions of the cells, avoiding illegal overlaps; (-) Proximity constraints barely supported and often weighted in the single-objective cost functions.			
Slicing	(+) The enhanced version in [40] presents the hierarchical and proximity constraint handling advantages of HB*-tree; (-) Not all floorplan topologies have a slicing structure, which degrade the density of the solutions.	[17, 40]	$O(\lg n)$	$O(n)$
Non-slicing	(+) No degradation of layout density			
Sequence pair [20]	(+) A placement configuration can be derived from any encoding.	[21-24]	$O(1)$	$O(n^2)$ [21-23] $O(G,n,\ln \ln n)$ [24]
BSG [25]	(+) More intuitive packing than the sequence pair (SP); (-) Cannot always represent the optimal packing for a determined group of cells; (-) Deficient support of constraints.	✗	n/d	$O(n^2)$
O-tree [26]	(+) Smaller complexity and less redundancy than SP/BSG, also fewer bits to describe the number of blocks; (-) Less flexible than SP and BSG in representation; (-) Tree structure is irregular, and thus some primitive tree operations (e.g., search, insertion) are less efficient.	[27]	$O(\lg n)$	$O(n^2)$
B*-tree [28]	(+) Upgrades O-tree in processing and efficiency; smaller encoding cost; no need for additional constraint graphs; (-) Less flexible than SP and BSG in representation.	[29-32]	$O(\lg n)$	$O(n \lg n)$
HB*-tree [37]	(+) Handle symmetry constraints in 2D with symmetry islands; possibility to combine symmetry islands with non-symmetric modules hierarchically; (+) Unlike remaining representations, can guarantee the closest proximity of symmetric modules within a symmetry group/island; (-) Less flexible than SP and BSG in representation.	✓	$O(\lg n)$	$O(n)$
TCG-S [34]	(+) Combine the advantages of SP, BSG and B*-tree; (+) Flexibility to handle placement with special constraints; (-) Despite improvements, the perturbation and packing complexity is still quadratic.	[36]	$O(n^2)$	$O(n^2)$

^aSymmetry-feasible conditions

^bPerturbation and packing complexity are related to the symmetry-feasible implementation of the structure, if any

The End