Ricardo Martins · Nuno Lourenço
Nuno Horta

Analog Integrated Circuit Design Automation

Placement, Routing and Parasitic Extraction Techniques

Springer

Analog Integrated Circuit Design Automation

Ricardo Martins • Nuno Lourenço • Nuno Horta

Analog Integrated Circuit Design Automation

Placement, Routing and Parasitic Extraction Techniques

Springer

Ricardo Martins
Instituto Superior Técnico,
Universidade de Lisboa
Instituto de Telecomunicações
Lisboa, Portugal

Nuno Horta
Instituto Superior Técnico,
Universidade de Lisboa
Instituto de Telecomunicações
Lisboa, Portugal

Nuno Lourenço
Instituto Superior Técnico,
Universidade de Lisboa
Instituto de Telecomunicações
Lisboa, Portugal

ISBN 978-3-319-34059-3 ISBN 978-3-319-34060-9 (eBook)
DOI 10.1007/978-3-319-34060-9

Library of Congress Control Number: 2016941743

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Ricardo Martins
To my little girls, Nádia, Joana, and Daniela
Nuno Lourenço
To Alina and Íris
Nuno Horta
To Carla, João, and Tiago

Preface

In the last years, the proliferation of consumer electronic devices triggered a huge increase in microelectronic activities, enabling the growth of the integrated circuit (IC) market from $10 billion in 1980 to over than $340 billion in 2015. Due to the developments made in terms of very-large-scale integration technologies, nowadays, designers have the means to build multimillion transistor ICs that implement complete systems in a single chip. The need of new functionalities, smaller devices, more power efficiency, less production and integration costs, and less design cost makes the design of electronic systems a truly challenging task, which must be completed within strict time-to-market constraints. Despite analog blocks representing only a small fraction of the die area, the effort in their design is considerably higher when compared to the effort spent on their digital counterpart, which is reflected in the total development time. Specifically, the development and test cost of analog or mixed-signal components usually surpasses the 50 % of total design cost, even though the area occupied can be as low as 3 % of the system-on-a-chip.

In digital IC design, several electronic design automation (EDA) tools and design methodologies are available to help designers in keeping up with new capabilities offered by state-of-the-art integration technologies, while analog design automation tools are not keeping up with those challenges. This difference in the level of automation happens because analog is, in general, less systematic, more heuristic and knowledge intensive than its digital counterpart, and becomes critical when digital and analog circuits are integrated together. Furthermore, while new fabrication technologies bring huge advantages to systems' performance, the reduction of devices' sizes and consequent increase in device density does not come only with benefits. This is especially true for the analog layout, where the layout induced disturbances, e.g., substrate noise, cross talk, supply noise, thermal noise, etc., effects became even more significant, having the potential to drastically affect the performance of the circuit. Moreover, the impact of layout-dependent effects, e.g., well proximity effect, poly spacing effect, length of diffusion, oxide to oxide spacing effect, etc., in deeper nanometer technologies can easily drive circuits to malfunction. Analog design automation has been intensively studied in academia for more than two decades, and although much has been accomplished, the fact is that there is still no

mature tool in the industrial environment and the analog layout is mostly done manually using time-consuming layout editors. Applications that provide some kind of user-assisted functionalities found their way into commercial EDA tools; however, the automatic functionalities are limited and lots of problems remain unsolved. For these reasons, the onset of more efficient and user-oriented tools to boost analog designers' productivity and ease this time-consuming task is mandatory.

The work presented in this book belongs to the scientific area of EDA and addresses the automatic generation of analog IC layout. A set of innovative placement, routing, and parasitic extraction methodologies for analog IC design automation were implemented in the tool AIDA-L. AIDA-L, which is integrated in the bottom-up physical synthesis path of an in-house analog IC design automation environment, AIDA, assists the analog designer in the iterative and error-prone process of layout generation. The designer specifies the circuit topology and the required technology and also provides intuitive high-level floorplan guidelines coded in a technology- and specification-independent template file. The high-level floorplan is mapped to a non-slicing topological representation, and the tool instantiates the devices and packs the floorplan for any set of devices' sizes provided. In the absence of the designer's guidelines, an innovative hierarchical multi-objective optimization, over an absolute representation, is used to provide a full Pareto set of placement solutions. For routing, the effects of current densities are considered to construct an electromigration-aware wiring topology for each power and signal network directly from the netlist, and then, several symmetry rules between wires are automatically identified and applied in the global and groundbreaking evolutionary detailed routing phases. Unlike previous approaches, the use of multiport structures for each terminal strongly enhances circuits' routability and the quality of the wiring symmetry. Furthermore, AIDA-L is also suited to the inclusion of layout-related data during automatic circuit sizing, by performing a fast, but accurate, 2.5-D parasitic extraction in loop over a semi-complete layout, whereas competing approaches require the complete detailed layout. The robustness of the automatic layout generation is demonstrated on several analog circuit structures, from simple amplifiers, generated in less than 1 min, to more complex circuits, generated in a few hours, using a 130 nm design process. The output layouts are stored in GDSII format and the results are validated, first, using an industrial-grade verification tool for design rule check and layout versus schematic and, then, with electric simulations over the extracted layouts.

Finally, the authors would like to express their gratitude for the financial support that made this work possible. The work developed in this book was supported in part by the Fundação para a Ciência e Tecnologia (grant FCT-SFRH/BD/86608/2012, grant FCT-SFRH/BPD/104648/2014, research project DISRUPTIVE EXCL/EEI-ELC/0261/2012, and research project UID/EEA/50008/2013) and by the Instituto de Telecomunicações (research project OPERA-PEst-OE/EEI/LA0008/2013).

This book is organized in nine chapters.

Chapter 1 presents a brief introduction to the area of analog IC design, with special emphasis to automatic layout generation. First, a well-accepted design flow for

analog ICs is presented, and then, AIDA-L's features and advances to the state of the art are outlined.

Chapter 2 presents a study of the available tools for analog layout design automation. The chapter starts by addressing the placement and routing problems in EDA, followed by the presentation of the main references of automatic layout generation tools and the most recent advances in analog layout-aware circuit sizing approaches. The available commercial solutions are also outlined throughout the chapter.

Chapter 3 gives an overview of the proposed automatic flow for analog IC design. Details about AIDA-L tool embedded in the AIDA's framework, as a stand-alone low-level layout generator and as part of layout-aware circuit sizing methodology, are presented. Additional detail about the tool's implementation, inputs, outputs, and interfaces is also provided.

Chapter 4 presents the methods used by the template-based Placer to process and place the modules in the floorplan, while following the high-level floorplan guidelines provided by the designer in the template file. Also, the B*-tree layout representation and all extraction and packing procedures are detailed.

Chapter 5 introduces the optimization-based Placer, where a well-known multi-objective evolutionary algorithm is enhanced and applied over the circuit's hierarchy. Instead of the template guidelines, current-flow and current-density considerations are taken, during an optimization over absolute coordinates, to improve the floorplan solutions.

Chapter 6 covers the description of the fully automatic Router architecture and generation procedure, depicting each task implemented in the AIDA-L's Router, i.e., the planning phases (electromigration-aware wiring topology construction and wiring symmetry detection), the global routing procedures (multiport selection and Steiner point assignment), and the evolutionary detailed routing phase.

Chapter 7 explains the methods used in the empirically based parasitic extractor to accurately compute the parasitic structures from a semi-complete layout. The processing of the intercap model tables is detailed, and also, the extraction of all resistive and capacitive structures.

Chapter 8 illustrates the application of the proposed design flow to practical examples on well-known analog circuit structures for a 130 nm design process. Also, placement and routing benchmark sets are used to evaluate each of the developed modules.

Chapter 9 shows the closing remarks, and the future directions for the continuous development of AIDA-L are outlined.

Lisboa, Portugal

Ricardo Martins
Nuno Lourenço
Nuno Horta

Contents

Abbreviations

AIDA	Analog IC Design Automation
AMG	Analog Module Generator
AMOSA	Archive-based Multi-objective Simulated Annealing
AMS	Analog and/or Mixed Signal
ASF	Automatically Symmetric Feasible
BSG	Bounded-Sliceline Grid
CAD	Computer-Aided Design
CAMOSA	Constrained Archive-based Multi-Objective Simulated Annealing
CMOS	Complementary Metal-Oxide-Semiconductor
DRC	Design Rule Check
EDA	Electronic Design Automation
EM	Electromigration
ERC	Electrical-Rule Check
GA	Genetic Algorithm
GDS	Graphic Database System
GUI	Graphical User Interface
HB*-tree	Hierarchical B*-Tree
HPWL	Half-Perimeter Wirelength
HS-tree	Hierarchical Slicing Tree
IC	Integrated Circuit
LDS	Layout Description Script
LP	Linear Programming
LVS	Layout Versus Schematic
MIM	Metal-Insulator-Metal
MOEA	Multi-Objective Evolutionary Algorithm
MOM	Metal-Oxide-Metal
MOO	Multi-Objective Optimization
MP	Multiport
MSP	Minimum Spanning Tree
MT	Multiterminal
MTF	Median Time to Failure

NP	Non-deterministic Polynomial-time
NSGA	Non-dominated Sorting Genetic Algorithm
O-Tree	Ordered Tree
POF	Pareto Optimal Front
PVT	Process, Voltage, and Temperature
RSMT	Rectilinear Steiner Minimal Tree
SA	Simulated Annealing
SMT	Steiner Minimal Tree
SO	Single-objective
SoC	System-on-a-Chip
SP	Symmetry Pair
TCG	Transitive Closure Graph
UMC	United Microelectronics Corporation
VLSI	Very-Large-Scale Integration
XML	Extensible Markup Language
WT	Wiring Topology
WS	Wiring Symmetry

Chapter 1
Introduction

In the last years, the proliferation of consumer electronic devices triggered a huge increase in microelectronic activities, enabling the growth of integrated circuits (ICs) market from $10 billion in 1980 to over than $340 billion in 2015, with the analog market growing 10.6 % in 2014, 0.7 points over the overall market growth, and expected to follow in 2015 ("World Semiconductor Trade Statistics (WSTS)," https://www.wsts.org/; McClean, "IC Market to Top $300 Billion for First Time in 2013," http://www.icinsights.com). The steady increase in overall performance of ICs has been mostly supported by an exponential growth in the density of transistors while inversely reducing the transistors' cost, as described by Moore's law (Proc. IEEE 86:82–85, 1998). Due to the developments made in the last decades in very large scale integration technologies, designers have the means to build extremely complex multimillion transistor ICs that implement complete systems in a single chip. The need of new functionalities, smaller devices, more power efficiency, less production and integration costs, and less design cost makes the design of electronic systems a truly challenging task, which must be completed within strict time-to-market constraints. The accomplishment of this complex task is only possible since designers are assisted by computer-aided design (CAD) tools that support the whole design process.

1.1 The AMS IC Design Flow

Even though most functions in today's ICs are implemented using digital or digital signal processing circuitry, analog circuits are the link between digital circuitry and the continuous-valued external world [1–3]. In modern telecommunications and multimedia applications the integration of complex systems-on-a-chip (SoC) comprising analog or mixed-signal (AMS) blocks together with digital processors and memory blocks [4, 5] is a common practice.

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_1

Due to the continuous nature of the signal values handled by analog circuits, they are much more sensitive to noise and process variations than their digital counterpart, leading to a more complex and time demanding design. Hence, designers have been replacing most of analog circuits by digital computations, however, the following list enumerates some typical blocks referred as remaining analog forever [6]:

- On the input side of a system, the signals from a sensor, microphone or antenna must be sensed or received, amplified and filtered up to a level that allows digitalization with satisfying signal-to-noise and distortion ratio. Typical application of these circuits is in sensor interfaces, telecommunication receivers or sound recording;
- On the output side of a system, the signal from digital processing must be reconverted to analog and it has to be strengthened, so that it can drive outside load with low distortion. These circuits are typically used in telecommunication transmitters and loudspeakers;
- Mixed-signal circuits like sample-and-hold, analog-to-digital converters and frequency synthesizers. These blocks establish the interface between input/output sides of a system and digital processing parts of a SoC;
- Voltage or current reference circuits, and also, crystal oscillators, to offer stable and absolute references for the above mentioned circuitry;
- The last block of analog circuits are the high-performance digital circuits. The prime example is state-of-the-art microprocessors that are customized like AMS circuits, attempting to reach higher speed and lower power consumption.

In terms of design flow specific for analog IC, it is acknowledged that each designer or company may have its own IC design flow. However, Gielen and Rutenbar [6] systematized all the steps that most designers take when manually designing an AMS IC, leading to the well-accepted design flow illustrated in Fig. 1.1. It consists of a series of top-down design steps repeated from the system-level to the device-level, and bottom-up layout generation and verification.

By adopting a hierarchical top-down design methodology is possible to perform system architectural exploration, obtaining a better overall system optimization at a higher abstraction level before starting more detailed implementations at the circuit- or device-level. Thus, attempting to find problems earlier in the AMS design flow increase the chances of first-time success, with fewer or no overall time consuming redesign iterations [4]. Nonetheless, the increased impact of layout parasitics and process variations with the state-of-the-art integration technologies is forcing many iterations in real world designs.

In this design flow the number of hierarchy levels depends on the complexity of the system being handled and there are no generally accepted representations for the architectural design, however, the steps between any two hierarchical levels are:

- Top-down electrical synthesis path, that includes topology selection, specification translation (or circuit sizing at the lowest level) and design verification;
- Bottom-up physical synthesis path, that includes layout generation and detailed design verification (after layout extraction).

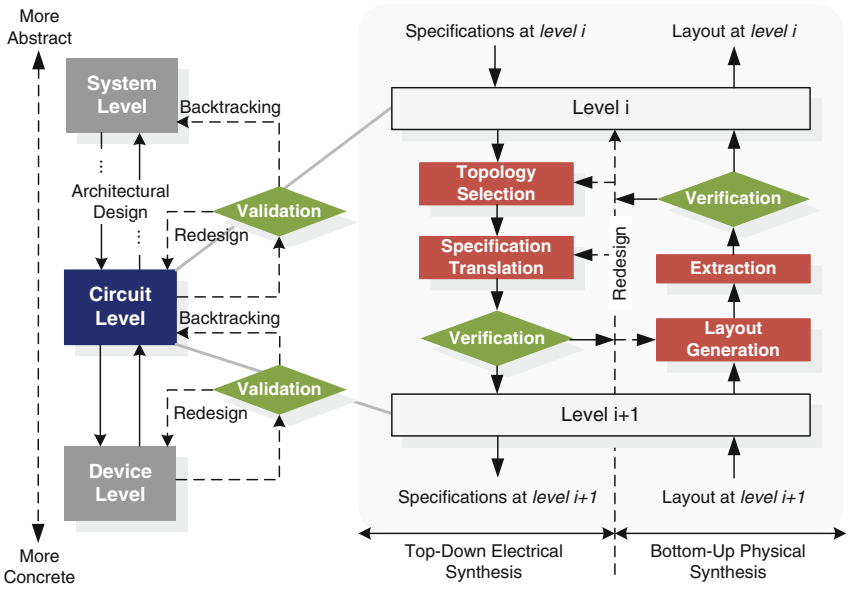


Fig. 1.1 Hierarchical levels and design tasks of AMS IC design process [6]

Topology selection is the step of determining the most appropriate system or circuit topology in order to meet a set of given specifications at the current hierarchy level. The topology can be either chosen from a set of available topologies, or synthetized.

Specification translation is the task of mapping the high-level block specifications, given a selected topology, into individual specifications for each of the sub-blocks. At the lowest level, the sub blocks are single devices and this task is reduced to circuit sizing. Specifications translation is verified by means of simulation before proceeding down the hierarchy. Since no device-level sizing is available at higher levels, simulations, at these levels, are behavioral. However, at the lowest levels in the design hierarchy, the circuit- and device-level, device sizing is available and, therefore, electrical simulations are used. The specifications for each of the blocks are passed to the next level of the hierarchy and the process is repeated until the top-down electrical synthesis flow is completed.

Several CAD tools, settled through the years in the industry, are fundamental to help the designer to successfully complete the circuit sizing task. They are used for IC design editing and evaluation, some of the tools available are: Mentor Graphics' ADiT, Questa and Eldo [7]; Synopsis' HSPICE, nanosim and HSim [8]; Cadence's Spectre [9]; ngspice [10]; and SMASH [11]. Also, some commercial automation solutions began to emerge as the result of the research efforts in this field, such as: Analog Design Automation Inc. Genius product line and Magma Titan that were integrated in Synopsys' frameworks in 2004 and 2012, respectively [8]; the circuit optimizer feature of Cadence's Virtuoso Custom Design Platform GXL [9]; Solido's

Fast process, voltage and temperature (PVT), Fast Monte Carlo and High-Sigma Monte Carlo [12]; or MunEDA's WiCkeD™ [13]. Even if applicable only at cell-level, i.e., generally analog components with 10–100 devices, they increase the automation level of the analog design environment.

From the bottom-up physical synthesis path, layout generation consists of creating the fabrication masks that are used to produce the devices. For the blocks at the lowest level in the design hierarchy this involves drawing the set of geometric shapes, which must obey strict design rules defined by the manufacturing process, that implement the intended devices and interconnections. For blocks at higher levels it involves placing and routing the layouts of the previously designed sub-blocks. In the provided design flow, is important to notice the presence of a detailed verification step over the extraction of the layout parasitics. In order to ascend to higher hierarchical levels is necessary that no potential problems are detected at the lowest levels and the circuit meets the target specifications after layout, i.e., post-layout verification. If not, backtracking and redesign steps are required. When the topmost level verification is complete, the system is designed and ready for fabrication.

Some mature CAD tools are available for layout edition, e.g., Mentor Graphics' IC Station Layout [7]; Synopsis' Galaxy Custom Designer LE [8]; and Cadence's Virtuoso Layout Editor [9]. Design rule and layout versus schematic (LVS) verification, and layout extraction can be performed, for example, in Mentor Graphics' CALIBRE [7]; Synopsis' Hercules [8]; and Cadence's DIVA and Assura [9].

1.2 Motivation for Analog Design Automation

In 1980, analog ICs represented 32 % of total IC shipments, that value increased to 49 % in 2010, and is forecast to grow to 57 % of total IC shipments by 2018 [14]. These numbers are supported by the rampant growth in Medical/Health, automotive, LED lighting and energy management for buildings that make extensive use of electronic devices where AMS blocks are integrated in SoC designs. Despite the fact that analog blocks constitute only a small fraction of the components on a mixed-signal IC and SoC design, the design effort of analog components is considerably higher when compared to the digital ones, which is reflected in the total development time. Specifically, the development and test cost of AMS components usually surpasses the 50 % of total design cost, even though the area occupied can be as low as 3 % of the total SoC [15]. The three main reasons pointed for the larger development cycle of analog blocks are: the lack of effective CAD tools for electronic design automation (EDA); analog circuits are being integrated using technologies optimized for digital circuits; and, analog blocks are difficult to design and reuse because they are more sensitive to surrounding circuitry, environmental and process variations than their digital counterpart.

As stated, today's analog design is supported by mature circuit simulators, layout editing environments and verification tools, however the design cycle for AMS ICs is still long and error-prone. These circuits suffer from diverse non-idealities and

parasitic disturbances that, by not being considered in the early stages of development, are responsible for design errors and expensive redesign cycles, making them the bottleneck of SoC designs. Given the giant growth of AMS market, the economic pressure for high-quality yet cheap electronic products and time-to-market constraints, there is great pressure for new EDA tools to facilitate the life of analog designers, increasing their productivity and simultaneously improve the quality of resulting ICs. In fact, all design tasks of the AMS IC flow presented in Fig. 1.1 have room for huge improvement in terms of user interaction, performance, and of course, automatic functionalities [15].

In the International Technology Roadmap for Semiconductors [5] the V-Cycle of a design summarizes the differences between analog and digital design automation, in particular, the implementation and the verification paths. The V-Cycle reveals that in the digital domain, EDA is fairly well developed and establish a low-level design process almost fully automated. The main gap in the digital design path of the system design are the tools and methodologies above the behavioral abstraction level. On the other hand, as the analog automation tools do not progress at the same pace of technology, knowledge and experience of the designer is always crucial for making decisions at all stages of the analog design flow. The "push-button" stage for analog design is still utopic and is historically a handcrafting process. EDA community has been primarily focused in the last decades on the evolution of the digital design automation methodologies to keep the pace with "More Moore" trend, featuring deep submicron technologies and multimillion transistor digital designs.

To conclude, in digital IC design several EDA tools and design methodologies are available to help the designers keeping up with the new capabilities offered by the state-of-the-art integration technologies, while analog design automation tools are not keeping up with the challenges created by technological evolution [16, 17]. This difference in the level of automation between analog and digital design is because analog is, in general, less systematic, more heuristic and knowledge intensive than the digital counterpart, and becomes critic when digital and analog circuits are integrated together. Due to the lack of automation, analog designers' knowledge and experience is crucial to manually explore the solution space searching for a solution that fulfills the design specifications, which allied with the non-reusable nature of the analog IC design, becomes a cumbersome task. Only in the recent years, EDA industry is turning its attention to analog design, due to its relevance in modern architectures.

Particularly, the layout generation task is pointed as the critical part of the automatic analog IC design flow [6, 18]. Despite the advantages that the new fabrication technologies bring to systems' performance, the huge increase in device density does not come only with benefits. This is especially true for analog layout, where the layout induced disturbances, crosstalk, substrate noise, supply noise, thermal noise, etc., effects became even more significant, having the potential to drastically affect the performance of the circuit. Moreover, as analog design enters in the deep nanometer era, with technology nodes at and below 40 nm, the impact of layout-dependent effects, e.g., well proximity effect, poly spacing effect, length of

diffusion, oxide to oxide spacing effect, etc., is becoming even more notorious, easily driving circuits to malfunction. As pointed by Steve Carlson [19], at 20-nm node, up to 30 % deviation of expected pre-layout performance can attributed only from the impact of layout design. Nonetheless, due to the difficulties found on this task, automated analog design tools should first concentrate on settling an analog-specific layout synthesis process and do not currently take into account all precision matching needs for such designs [5].

1.3 Analog Layout Automation

The methodologies proposed and presented in this book focus on the layout generation task of analog ICs. Analog design automation has been intensively studied in academia for more than two decades (as overviewed later in Chap. 2 of this book) and is still an intensive research topic [20–22, 26]. Although much has been accomplished, the fact is that there is still no mature tool in the industrial environment and the analog layout is mostly done manually using time-consuming layout editors. Applications that provide some kind of user-assisted functionalities found their way into commercial EDA tools, however, the automatic functionalities are limited, far from perfect and lots of problems remain unsolved [16]. Allied to the fact that these functionalities are constantly discredited by analog designers, EDA vendors do not put enough effort on their continuous development and improvement. The onset of more efficient and user-oriented tools is mandatory in order to boost analog designers' productivity and ease this time-consuming task.

In the traditional design flow, layout generation is only triggered when circuit sizing is complete. However, to achieve post-layout successful designs that meet all specifications, time-consuming and non-systematic iterations between these electrical and physical design phases are required. Without them, performance overdesign results in wasted power and area, and if underestimated, the circuits' post-layout performance can be compromised [18, 23]. Automatic circuit sizing methodologies in both research environment as in the industry are way more developed than layout generators, and the urge for the so called layout-aware or layout-driven methodologies (i.e., that include layout-related data, e.g., geometrical information from the layout or parasitic components, into the sizing task) to close the gap between electrical and physical design steps, really enforces the need of automatic procedures to generate the layout. Fast, flexible and as robust as possible layout generators are mandatory to include precise layout-related data into the sizing process, and, eventually, obtain a final layout simultaneously with sizing.

Generally the complexity of designing analog circuits' layout is not due to the number of devices, but from the countless interactions between them. Plus, for smaller technology nodes with the increased complexity of the design rules and physical effects that impact those interactions strongly, complexity is even greater. At cell-level, the automatic floorplan generation methodologies although far from the "push-button" stage, are keeping up relatively well with the design challenges

imposed by the integration technologies, although, being overlooked by the majority of the analog designers. However, in the automatic routing generation, most of the difficulties found in the initial approaches remain the same.

To summarize these introductory paragraphs, the precise goals of the proposed automatic layout generator tool are:

- *To be embedded in a layout-aware circuit sizing methodology*: develop a computationally efficient complete layout generation tool that can be embedded on an automatic IC design flow, and therefore feed layout-related data, both geometrical information and parasitic components, to the automatic sizing task. Currently, most layout generators integrated in layout-aware methodologies rely on parametric cells (i.e., code the entire layout of a circuit in a software tool) which are fast but lack of flexibility and demand huge setup effort. On the other hand, fully-automatic layout generation and accurate parasitic extraction are excessively time consuming and hard to setup operations, to be included in the sizing loop.
- *To be used as standalone and provide a fully functional first cut design*: analog designers' productivity is increased by automatically generating layouts for analog cells. The GDSII description of these cells must be validated, in a first phase, with an industrial grade verification tool, e.g., Mentor Graphics' Calibre®, and successfully pass design rule check (DRC) and LVS. After, the specifications of the original design must be validated with electrical simulations over the extracted netlist, in the presence of complete layout parasitics. The goal is to provide designers with a fully functional first cut layout design, aware that analog layout design is extremely subject to the aesthetic engineering.

The majority of the designers and EDA developers adopt a layout generation flow that, traditionally, starts with placement (where the devices are positioned in the chip area), and only after, routing (where the interconnections between them are made). However, placement and routing tasks tend to overlap in order to consider routing-related data earlier in the design flow. For these two tasks, specifically:

- *To support both user-assisted and fully-automatic placement generation*: in the user-assisted aspect, provide an intuitive, easy, technology- and specification-independent way for the designer to include his knowledge into the generation process. The objective is not to replace the designer in the whole generation task, but rather use his guidelines as starting point to rapidly obtain a solution that can be used as a first cut design. Handmade designs are known for their robustness, and the tool should provide the means for the designer easily integrate his knowledge and lighten the efforts to accomplish placement task, abstractly from the technological details. However, in the absence of these designer's guidelines, the tool should be capable of placing the devices and present a satisfactory floorplan solution, or, a set of solutions for the designer to choose from.
- *To alleviate the analog designer from the time-consuming routing task*: the quality of the routing is strongly dependent of the quality of the floorplan that is used as starting point. It is known that any change in the floorplan may require a complete routing redesign, so, the objective is to automatically generate the routing from the exactly same setup no matter which is the floorplan provided. Semi-automatic

Routers were actually very unsuccessful in the past, mostly because the results do not fully meet analog designers' expectations. This goal will require deep understanding of the previous approaches and, furthermore, research and development of different techniques to progress beyond the results of the existent solutions.

1.4 Advances to the State-of-the-Art

The main innovative contributions of the methodologies proposed in this book were integrated in the AIDA framework [24, 25] and can be summarized as follows:

- *Hierarchical combination of Pareto fronts of placements*: analog IC floorplan automation is complex as multiple requirements, which appear mainly in the form of topological constraints, must be dealt simultaneously along with several objectives and/or considerations for a robust floorplan. Absolute coordinates is the most practical and intuitive manner of implementing those layout constraints/requirements. However, a complete study of previous absolute floorplanners suggests that illegal overlaps and other constraints have been improperly weighted in a single-objective cost function for optimization along with other objectives. The problem of analog floorplan automation in absolute coordinates is here reformulated, and, since it is impossible to determine a single best floorplan for all of the design objectives, a constrained multi-objective optimization algorithm is proposed to solve it. In order to reduce the problem's complexity, the concept of proximity group is introduced, and the Pareto fronts of placements representing the tradeoffs between the optimization objectives for each group are combined bottom-up through the design hierarchy. Current-flow and current-density considerations are taken during the optimization to improve routing quality, reliability and, attempting to reduce routing-induced parasitics for better post-layout circuit performance.
- *Electromigration-aware routing with multilayer multiport terminal structures*: in real analog cells, a complicated terminal geometry can easily have tens of ports (electrically equivalent locations) on multiple fabrication layers that are consider in manual design and, should also be considered in automatic approaches to increase routing efficiency. However, in the state-of-the-art of analog layout automation only simplistic single-ports/'dot-models' to represent these complex multiport structures are considered. Instead of assuming that an unknown shape of a group of ports is an element of a 'dot-model', in the proposed methodology the solution is found considering every port available, over different fabrication layers, of the terminal geometry, which further increases the complexity of the original NP-complete Steiner problem. In addition, electromigration and voltage drop in the wires are taken in consideration, without adding setup complexity. The Router module inputs only the netlist and the electric-currents for each terminal and introduces a new degree of freedom by automatically exploring all the available electrically-equivalent ports of a terminal to connect a wire, in both

global and detailed routing phases. No additional setup/tuning is required for a symmetric electromigration- and IR-drop-reliable solution.

- *Evolutionary multi-objective multi-constraint detailed Router*: unlike the available deterministic approaches, wires are not restricted to limitative representations and are represented by their absolute coordinates, which expand the effectiveness of the exploration of the solution space. An in-loop internal procedure is used to evaluate each generation of layout solutions, instead of forcing the design rules in the path-finding algorithms by expanding the grids or routing channels with the minimum space requirements. This robust but lightweight built-in layout evaluation procedure is compliant with industrial grade validation tools. All nets are optimized simultaneously to achieve a solution in the conditions imposed by the problem definition, which eliminates the need for deterministic and error-prone backtracking, rip-up, re-routing and net-ordering considerations.
- *Parasitic extraction performed over a semi-complete layout*: to address post-layout performance degradation and geometric requirements earlier in the design flow, the layout-aware design approaches include layout effects during the automatic sizing loop. However, both complete automatic layout generation and exhaustive parasitic extraction are still time consuming and hard to setup operations. By using a lightweight built-in extractor it is possible to accurately compute the impact of layout parasitics for both floorplan and early-stages of routing in-loop, without requiring a detailed and final layout (i.e., a layout with all DRC and LVS errors solved), unlike previous approaches. Traditionally, the last step required in the automatic layout generation flow is the detailed routing, which is by far the most computational intensive and time consuming step. By avoiding the need of a detailed layout and consequently an external extractor, the overall optimization time is greatly reduced.

1.5 Conclusion

While in the digital design several EDA tools are available to help the designers keeping up with the new capabilities offered by the integration technologies, analog IC design automation tools are not keeping up with the challenges created by the technological evolution. This is just one of the reasons why analog design is many technology nodes behind leading-edge digital. Therefore, due to the lack of automation, analog designers keep exploring manually the solution space, searching for a solution that fulfills all the design specifications. This method causes long design time and, allied to the complexity and non-reusable nature of analog IC design, makes it a cumbersome task. However, after many years of stagnation due to heavy investment in the digital domain, the once-sleepy analog design automation market is now evolving. Simultaneously, in this Chapter, the objectives and contributions of the layout generation tool proposed in this book were outlined, that aim to ease the efforts of analog designers in this time-consuming task.

References

1. World Semiconductor Trade Statistics (WSTS), https://www.wsts.org/
2. B. McClean, IC market to top $300 billion for first time in 2013, http://www.icinsights.com
3. G. Moore, Cramming more components onto integrated circuits. Proc. IEEE **86**(1), 82–85 (1998)
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. Mentor Graphics, http://www.mentor.com/
8. Synopsis, http://www.synopsys.com
9. Cadence Design Systems Inc, http://www.cadence.com
10. gEDA project, http://www.gpleda.org
11. Dolphin Integration, http://www.dolphin.fr
12. Solido Design Automation, http://www.solidodesign.com/
13. MunEda, http://www.muneda.com/
14. IC Insights, Inc. Analog unit shipments outpacing growth of all IC product segments (2014), http://www.icinsights.com/data/articles/documents/705.pdf
15. 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
16. R.A. Rutenbar, Analog layout synthesis: What's missing? in *Proc. ACM/SIGDA International Symposium on Physical Design (ISPD)*, Jan 2010, p. 43
17. G. Gielen, E. Maricau, P. Wit, Design automation towards reliable analog integrated circuits, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2010, pp. 248–251
18. H.E. Graeb (ed.), *Analog layout synthesis: A survey of topological approaches* (Springer, New York, 2010)
19. S. Carlson, The five key challenges of sub-28 nm custom and analog design (2013), http://www.techdesignforums.com/practice/technique/five-key-challenges-20nm-custom-design/
20. H.E. Graeb, ITRS 2011 analog EDA challenges and approaches, in *Proceedings on Design, Automation & Test in Europe (DATE)*, Mar 2012, pp. 1150–1155.
21. 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. Circ. Syst. **34**(2), 199–212 (2015)
22. X. Dong, L. Zhang, Lithography-aware analog layout retargeting. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **35**(2), 232–245 (2016)
23. R. Castro-Lopez, O. Guerra, E. Roca, F. Fernandez, An integrated layout-synthesis approach for analog ICs. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **27**(7), 1179–1189 (2008)
24. N. Lourenço, R. Martins, N. Horta, Layout-aware synthesis of analog ICs using flooplan & routing estimates for parasitic extraction, in *Design, Automation & Test in Europe Conference (DATE)*, Mar 2015, pp. 1156–1161
25. R. Martins, N. Lourenço, A. Canelas, R. Póvoa, N. Horta, AIDA: Robust layout-aware synthesis of analog ICs including sizing and layout, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sept 2015, pp. 1–4
26. M.P.-H. Lin, Y.-W. Chang and C.-M. Hung, Recent research development and new challenges in analog layout synthesis, in 21st Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 617–622, January 2016

Chapter 2
State-of-the-Art on Analog Layout Automation

In the past few years, several tools for the automation of analog integrated circuit (IC) layout design with application on both new and reused designs have emerged. Yet, in IC industry, most of the analog layout design is still handmade, essentially because analog designers want to have total control over the design options and, the fact that current fully automated analog layout generators produce solutions that do not fully meet analog designers' expectations and often are not yet competitive with manually crafted ones. The state-of-the-art on analog layout automation presented in this Chapter reveals that after many years of stagnation, electronic design automation (EDA) ecosystem is evolving, creating more robust, efficient and complementary approaches to the existing tools.

Having the devices, for the selected topology, sized, they must be laid out in the chip; a common analog layout design approach is to split the problem into two problems, placement and routing. Following this approach, this Chapter starts by addressing the placement problem in EDA, providing a brief overview of the different available floorplan representations and most recent challenges presented to placement tools. In section "Routing", an overview of existent routing algorithms is presented, with emphasis on electromigration-aware approaches and wiring symmetry considerations. In section "Complete Layout Generation Tools", the main references of complete automatic layout generators are presented, and, in section "Closing the Gap Between Electrical and Physical Design", the analog circuit-sizing task is overviewed to present the most recent advances in automatic layout-aware circuit-sizing approaches. Finally, in section "Overview of the State-of-the-Art on Analog Layout Automation", a global analysis of the state-of-the-art is made as introduction for the implementation choices taken for the methodologies proposed in this book.

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_2

2.1 Placement

An automatic placement tool should produce solutions similar in density and performance to the high-quality manual layouts. However, to reduce the unwanted impact of layout parasitic structures, process variations and different on-die operating conditions, placement of analog devices must be made taking simultaneously many requirements into consideration. Those requirements, which traditionally appear in the form of symmetry, matching and proximity constraints, allied to the multitude of possible device implementations, with different sizes and aspect ratios, make the analog placement task hard to automate. Furthermore, these constraints must be strictly satisfied while attempting to minimize several objectives, such as chip area, interconnect length or parasitic impact. This is fundamental as the attainable routing quality and, most of the parasitic effects and consequent post-layout circuit's performance degradation are set once a placement solution is fixed.

2.1.1 Analog Topological Constraints

The major topological constraints for analog placement are device matching, device symmetry and device proximity [1], as illustrated in Fig. 2.1. Symmetry restricts devices to a mirrored placement, it is used to offset geometric and electrical issues, and helps reducing the sensitivity to on-die thermal gradients and parasitic mismatches between two identical signal flows. Matching forces a common gate orientation, common centroid or an interdigitized placement among devices, which improves the beneficial effects of symmetry by reducing even further the impact of process-induced mismatches on the devices. Proximity limits devices to a specific placement so they can share a common substrate/well region, be surrounded by a common guard ring, being also used to place closely matched/symmetrical devices. By ensuring on-die proximity, the effect of substrate coupling is decreased, and also, avoids large mismatch and deviations during the fabrication process [1, 2]. Variations of proximity constraints for maximizing distance between modules are also common on analog layout design.

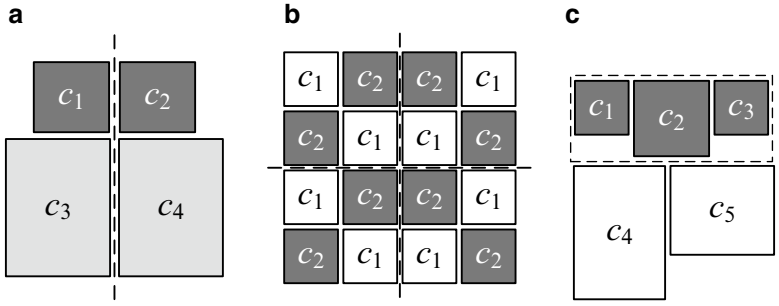


Fig. 2.1 Representation of topological constraints for analog layout design: (**a**) Symmetry; (**b**) Matching (common centroid); and (**c**) Proximity (guard ring/well)

2.1.2 Floorplan Representations

One of the most relevant factors when developing a placement tool is its representation of the cells and each placement tool has its own strategy for it. The two main classes of approaches that have been used in the last years are distinguished by how the optimizer encodes and moves the cells [1]: by absolute representation, i.e., cells are represented by means of absolute coordinates, and by means of a relative representation, i.e., encoding the positioning relations between any pair of cells, the last one is further classified into slicing or non-slicing representations.

2.1.2.1 Absolute Representation

In the absolute representation the optimizer moves the cells explicitly, as each cell is represented by means of absolute coordinates, i.e., the manufacturing grid, in this way, every possible placement can be described. It proved to be the most practical solution to implement topological constraints since they can be minimized or forced using the coordinates directly. However, this type of representation allows illegal overlaps during the moves (e.g., cell translation or rotation) and, since no restriction is made referring to the relative position of a cell with respect to another cell (except in some cases for symmetry pairs) a huge search space of both feasible and unfeasible solutions needs to be explored.

The first use of absolute coordinates is reported in [3]. Applied to analog IC design, in KOAN/ANAGRAM II [4] the simulated-annealing (SA)-based kernel [5], which is detailed in Sect. 2.1.4, is used to minimize the cost function:

$$\min f(x) = \alpha_1 f_1 + \alpha_2 f_2(x) + \sum_i \alpha_i f_i(x) \tag{2.1}$$

where x is the design variables vector (i.e., the coordinates and, if allowed, the rotation of each cell) and α is the weights vector. $f_1(x)$ and $f_2(x)$ are the two primarily/fundamental objectives that need to be minimized, namely, placement area and overlaps (whose penalty cost must not only be minimized but driven to zero in the final solution, i.e., $f_2(x)=0$, to attain an admissible placement) between cells. Other $f_i(x)$ objectives are minimized simultaneously in the single-objective cost function, e.g., aspect ratio of the floorplan, estimation of net length (by means of a pseudo-Spanning tree), proximity and/or merge considerations.

In LAYLA [6], PUPPY-A [7] and ALDAC [8] the same scheme of SA-based minimization of the cost function (2.1) is used. Some variants of the $f_i(x)$ objectives are considered, such as net length estimation by means of half perimeter or pseudo-Steiner tree [7] or some heuristics to predict the performance degradation [6, 7]. The weights α are experimentally chosen nonnegative values at priori or adapted dynamically during the placement optimization. In the dynamic approach, area and wire length are usually set to dominate the cost function initially and their importance decreases progressively. At low temperatures of the SA kernel, overlaps and symmetries (if not imposed) become the dominant parameters [7]. However, the need of a

constant tuning effort due to the difficulty on predicting an appropriate weight for the overlap penalty is one of the main disadvantages of using an absolute representation.

Some attempts to pure genetic algorithm (GA)-based optimization [9] to analog placement problems were made [10–12], which show a fast convergence rate in the initial phase, however, the degrading of search efficiency due to the GA incapability of accepting unfavorable solutions resulted in premature/sparse and sub-optimal floorplans. Unlike previous approaches, in [13] a GA-based optimization whose mutation is controlled by SA to improve diversity is used. In order to avoid unfeasible solutions, a deterministic cell slide post-processing technique is performed after each mutation that transforms the absolute placement into a relative one, solving the overlaps and forcing the desired symmetries. However, it requires determining the relative positions of macrocells after they have been placed by absolute coordinates. The several objectives are again attained by minimizing a weighted single-objective cost function.

2.1.2.2 Relative Representation

In relative representations, the optimizer does not move cells explicitly; instead, it changes the relative position of cells by perturbing the structure that encodes the floorplan. Then, a packing procedure is performed to transform the relative representation to a floorplan while avoiding overlaps and implementing the supported constraints. The solution space is greatly reduced as only feasible regions (admissible placements) of the solution space are explored. However, depending on the topological representation used, the optimal solution can be left out of the search space. Furthermore, since the optimization kernel only changes the relative positioning, symmetry-feasible conditions by means of structure scan or post-processing in order to penalize, avoid or fix the symmetries violated each time the typically SA-based kernel perturbs the structure must be derived for each representation.

Slicing

The first class of relative representations is the slicing model, where cells are organized in sets of slices that recursively bisect the layout horizontally and vertically. The direction and nesting of the slices can be recorded in a slicing tree or, equivalently, in a normalized Polish expression [14]. Figure 2.2 represents a slicing structure, which is obtained by recursively cut rectangles into smaller ones, the corresponding slicing tree is also presented. The internal nodes marked with "*" represent vertical cuts, and the ones marked with "+" represent horizontal cuts. Since not all the layout topologies have a slicing structure, this representation can degrade the density of the placement solution. This is especially true when the layout's cells are very different in aspect ratio, a common situation in analog circuits.

More recently, the slicing-tree representation has been popular in methodologies for layout migration [15] and for fixed-outline floorplanning [16], however, only in [17] symmetric-feasibility conditions for the slicing-tree were introduced.

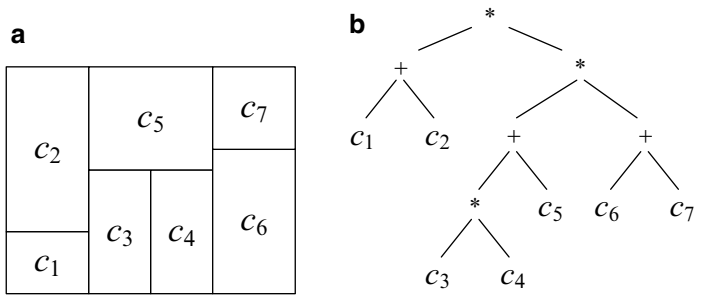
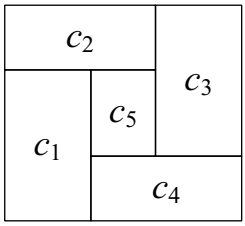


Fig. 2.2 Slicing structure (**a**) and correspondent slicing tree (**b**) example [14]

Fig. 2.3 Non-slicing structure



The weaknesses identified in the slicing model culminated in the emergence of several non-slicing relative representations, for these models the degradation of layout density is no longer a matter of concern [18]. However, and to be fair, the slicing tree can generate non-slicing placements if followed by compaction [19].

Non-slicing

The non-slicing floorplan is more general than the slicing floorplan, an example of a non-slicing structure is presented in Fig. 2.3.

There are several non-slicing representations available nowadays. The sequence-pair encodes the "left-right" and "up-down" positioning relations between cells [20], and, the solution space can be effectively explored as a placement configuration can be derived from any encoding. Symmetry constraints can be handled with an $O(n^2)$ packing complexity, where n is the number of placeable cells, as described in [21]. Koda et al. [22] also attains the same packing complexity by using linear programming during placement process and in [23] by enhancing the sequence-pair with dummy nodes. In [24], a priority queue allows to pack the sequence-pair with only an $O(G.n.\log(\log(n)))$ complexity, where G is the number of symmetry groups, i.e., a symmetry group is subset of cells that share a common symmetry axis.

The bounded-sliceline grid (BSG) [25] also has an $O(n^2)$ packing complexity; it uses a meta-grid structure without physical dimensions, but introduces orthogonal unique relations of "right-of" and "above" for each pair of cells. It poses a more

intuitive packing than the sequence-pair, however, the BSG may be incapable of representing the optimal packing for a given problem and its support of symmetry constraints has not been proved yet.

The ordered tree (O-tree) [26] extended the slicing tree to the representation of non-slicing structures, in addition, it shows a complexity even smaller than the original slicing tree. This method was introduced to reduce the drawback of redundancies from sequence-pair and BSG representations, and also, it needs fewer bits to describe the same number of blocks than those methods. The complexity for transforming O-tree, without symmetries, to its representing placement is linear to the number of blocks, $O(n)$. Symmetry-feasible conditions for the O-tree were introduced in [27].

An upgraded tree representation, B*-tree [28], is also available. It offers a $O(n.\log(n))$ packing complexity for a binary tree structure and do not requires additional constraint graphs for cost computation, while the other methods presented above require them. Unlike O-tree, the root of the ordered binary tree corresponds to the block at the bottom left corner of the floorplan. Several symmetry-feasible conditions by means of enhancement with segment trees [29, 30], red–black interval trees [31] or skip list [32] were developed for the B*-tree. In Fig. 2.4 an example of a placement with two symmetry constraints is presented and their respective representation in sequence-pair, O-tree and B*-tree encodings.

The transitive closure graph-based (TCG) [33] representation was proposed to combine the advantages of sequence-pair, BSG and B*-tree representations, guaranteeing a unique feasible packing for each representation and that no additional constraint graphs are constructed for the cost evaluation during packing. This approach

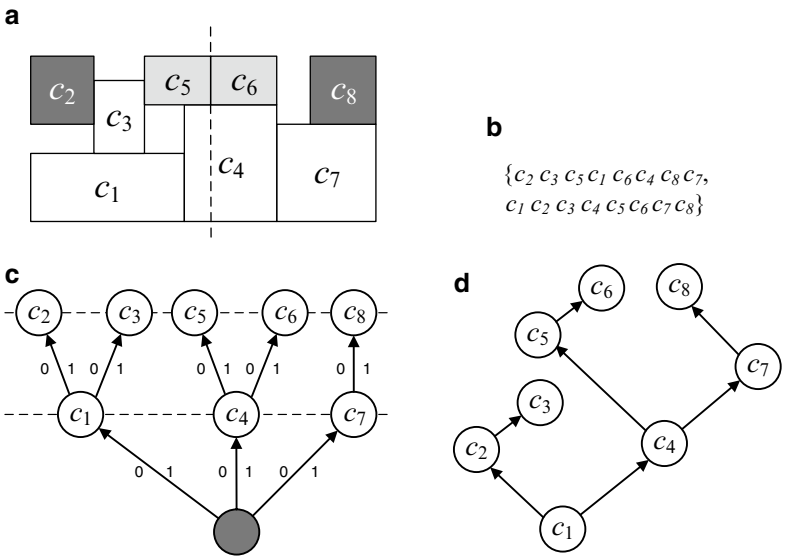


Fig. 2.4 Example of topological layout representations [29]: (**a**) Placement with two symmetry pairs; (**b**) sequence-pair encoding; (**c**) O-tree encoding; and (**d**) B*-tree encoding

was replaced by TCG-S [34], derived from the TCG combined with sequence-pair, it presents the fast packing characteristic of sequence-pair while maintaining the TCG flexibility to handle placement with special constraints. Symmetry-feasible conditions for the TCG and TCG-S can be found in [35] and [36], respectively.

More recently, Lin et al. [37–39] introduced the concept of symmetry island, which is the only relative representation that deals with proximity constraints efficiently by keeping modules of the same symmetry group (symmetry pairs and/or self-symmetric cells), i.e., that share the same symmetry axis, close to each other. To model a specific placement within a symmetry island a structure based on the B*-tree is used, called automatically symmetric feasible (ASF) B*-tree. The principal task of this algorithm is performed on a structure, a hierarchical B*-tree (HB*-tree), to simultaneously optimize the placement with both symmetry islands and non-symmetry modules, and dynamically update the shape for the devices within symmetry islands. The same hierarchical framework with symmetry islands was recently applied over a Slicing-tree representation [40]. These works present interesting features for hierarchical symmetry, hierarchical proximity and device clustering, mandatory requirements in modern analog layout designs.

In Table 2.1 a summary of the advantages, drawbacks, and perturbation and packing complexities identified for each of the above representations is presented.

2.1.3 Challenges in Modern Analog Placement

Over the years different methodologies have explored the advantages of the overviewed floorplan representations and new ways of treating layout constraints. These tools had been integrated with more or less success in analog generation tools. However, modern design challenges, e.g., thermal effects, sensitivity of signal-paths and current-paths, impact of layout parasitics on circuit's performance, etc., have reshaped the placement problem beyond the typical problem of area and/or wirelength minimization under several constraints [41].

2.1.3.1 Pareto Front of Placement Solutions

Due to the multiple objectives that need to be considered simultaneously, it may be impossible to decide the single best placement for a given problem. In a time dominated by optimization algorithms a fully deterministic approach aroused, Plantage [2], which is based on hierarchically bounded enumeration of basic building blocks using B*-trees. This approach is based on the principle that analog circuits naturally show a hierarchical structure, so, hierarchy is used to bound the enumeration, aware that the complete enumeration of all possible placements is impracticable.

The algorithm starts by generating all placements of the basic modules (leaf nodes of the hierarchy tree) and then the results of the enumerations are combined bottom-up, guided by the hierarchy tree. The process is carried until a Pareto front

Table 2.1 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	Slicing tree/ polish [14]	(+) 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]	$O(n^2)$	$O(n^2)$

[a]Symmetry-feasible conditions
[b]Perturbation and packing complexity are related to the symmetry-feasible implementation of the structure, if any

of placements with different aspect ratios for the whole circuit, resultant from the exploration of the tradeoff placements' width versus height, is obtained. Enhanced shape functions are used to store and combine modules efficiently, these functions consist of an ordered set of shapes which are classified through the process by aspect ratio and redundancy, and also, modules considered suboptimal are removed for the sake of computational effort.

2.1.3.2 Thermal-Driven Placement

In a different direction from the other emerging works, Lin et al. [42–44] proposed a thermal-driven analog placement solution, that simultaneously optimize the placement of power and non-power (devices that consume much less power than those classified as power) devices, in an attempt to annihilate thermally-induced mismatches. The thermal impact from power devices can affect the electrical characteristics of the other thermally-sensitive modules, degrading analog or mixed-signal (AMS) ICs performance. A thermal profile for a given circuit for better thermal matching of the devices is established, and the algorithm evolves until the desired thermal profile is achieved, i.e., the even distribution of the heat for the whole chip.

2.1.3.3 Current-Driven Placement

As the attainable routing quality, as well as, most parasitic effects and consequent post-layout circuit's performance degradation are set once a placement solution is fixed, current-flow and current-density considerations were recently brought into the placement automation [45]. Smooth signal-flows and current-flows lead not only to reduced layout parasitics but also smaller mismatch, while current-density constraints are used to define the width of the net's wires to avoid IR-drop and/or electromigration-related problems.

The first approach that considers current-flow constraints during automatic placement is found in [46]. There, the modules are placed closely in a predefined straight line according to a sequence of current-flow constraints, and then, the dimensions of the MOS transistors are adapted attempting to improve the quality of the design. In [40, 47] the deterministic algorithm DeFer [16] operates over a hierarchical slicing-tree to provide solutions with monotonic current-paths. The half-perimeter wirelength (HPWL) is optimized using linear programming when all the topological and current-path constraints are satisfied. In [45] both current-flow and current-density constraints are addressed at the placement stage by modeling cell's devices and interconnects in an enhanced HB*-tree representation. A constructive dynamic programming-based global router simultaneously place and route the modules, and, reserves routing space between cells.

2.1.4 Optimization Algorithm of Choice: Simulated Annealing

SA [5] is one of the most popular stochastic techniques, and is used in most of the optimization-based floorplan approaches for analog design automation proposed in the last decades. SA is a stochastic point-to-point search method that draws analogy from annealing of solids. Without the loss of generality it is assumed that in a minimization problem the current solution u is moved to a neighbor v, which can be accepted according to the probability $p(\mathrm{f}(u), \mathrm{f}(v))$:

$$p\left(f\left(u\right), f\left(v\right)\right) = \frac{1}{1 + e^{\left\lceil \left(f(u) - f(v)\right)/\left(T * f(u)\right)\right\rceil}} \tag{2.2}$$

where $\mathrm{f}(u)$ and $\mathrm{f}(v)$ are the relative performances of the current and neighbor solutions. The probability can be a more flat or a more abrupt function of the relative performance, this is controlled with the parameter T, which commonly scheduled as an exponentially decreasing function of time:

$$T\left(t\right) = T_{\max} \cdot e^{-R \cdot \left(\frac{t}{k}\right)} \tag{2.3}$$

where R is the temperature decreasing rate, k a scale factor for the iteration counter t and T_{max} is the initial temperature. For high values of T, the probability p gets flat around 50 %, i.e., explore the solution space, and for low values of T, the probability p can be approximately given by Eq. (2.4), i.e., exploit.

$$p\left(f\left(u\right), f\left(v\right)\right) = \begin{cases} 1 \text{ when } f(v) < f(u) \\ 0 \text{ when } f(v) \geq f(u) \end{cases} \tag{2.4}$$

This way, SA is capable of accepting unfavorable solutions probabilistically. The use of stochastically controlled hill-climbing allows SA to avoid local minima during the optimization process.

2.1.5 Commercial Solutions

Recently, some commercial solutions have emerged in the analog layout EDA market. Tanner EDA [48] HiPer DevGen, acquired by Mentor Graphics'®, presents a smart generator to accelerate the creation of standard cells. The tool analyzes the netlist and recognizes the current mirrors and differential pairs, and then automatically sends them to the generators. The generated primitives intend to be similar to those handcrafted, where designers have the control over the generation options, i.e., placement and routing of these structures. For differential pairs there are multiple options to ensure matching, optimized parasitic, add dummy devices, guard rings, antenna effect diodes, etc. For current mirrors there are multiple outputs of different electric-current strengths, options to ensure matching, add dummy devices,

share diffusion, multiple finger with options for gate and bulk connections, and adjustments for well proximity effects. To configure a new technology only the design rules are required to have design-rule correct and layout-versus-schematic verified standard cells.

Synopsys® Helix™ [49] is a placement manager supported by a powerful and easy to use graphical user interface (GUI). The designer introduces the system hierarchy and each of the sub-blocks can be added independently from the remaining. This perspective is useful on an on-going system-level specifications translation, since the parasitics of the available blocks and estimated areas can be provided for the designer to optimize the system-level design. For the automatic placement, the designer provides a set of constraints for a given circuit schematic and the tool automatically presents a set of possible minimum-spacing layout alternatives for that block. The tool explores the possible combinations deterministically and produces design-rule correct placement solutions for modern technology design processes. The output is a standard OpenAccess database that can be edited in most of the layout editors.

2.2 Routing

In the traditional analog design flow, the routing task is usually only triggered when the placement task is complete. As stated, automatic floorplan generators, although far from the "push-button" stage, are keeping up relatively well with the challenges imposed by the integration technologies, however, most of the difficulties found in the initial automatic routing approaches remain the same.

2.2.1 From Netlist to Pathfinding

While the netlist is available for any circuit, it does not provide any information on how the devices' terminals should be connect, that is, the terminal-to-terminal connectivity is also needed. Starting from the netlist, if the terminal-to-terminal connectivity of each net is unknown, the routing of single-port multiterminal signal nets is usually addressed as the classical Steiner minimal tree (SMT) problem. Where, a set of Steiner points must be found in order to minimize the SMT total interconnect length that contains all the terminals of the net. This problem is often generalized as the rectilinear Steiner minimal tree (RSMT), when only edges defined by vertical or horizontal segments are considered, however, the problem is still NP-complete [50].

A RSMT global router is used in [51] for wiring length estimation, and, when the terminal-to-terminal connectivity is found, the traditional pathfinding algorithm is used. These deterministic pathfinding algorithms developed in the late 1980s, are variations of the classic maze algorithm [51–53], which is the most common approach, but line-expansion techniques [4] can also be found. Each instance of

those approaches is used to generate a wire that connects two different terminals in the presence of obstacles (e.g., devices placed on the floorplan or other wires), usually by means of a grid-based or tile-based representation to ensure no overlaps or design rule violations, where the routing of all nets is done by iterating the different wires. The design rule validations are usually forced in the path-finding algorithms, e.g., by expanding the grids or routing channels with the minimum space requirements. Since an analog cell has a considerable number of conflicting nets, each one containing multiple wires, heuristics for net (re)ordering, backtracking and re-routing must be used to obtain valid solutions [1].

A different approach to automatic routing is the template adjustment techniques [54, 55], these have the highest setup times, but outperform the remaining by its fast and user-defined generation. A detailed state-of-the-art on the pathfinding algorithms and analog layout routing difficulties can be found in the Chapter *Routing Analog Circuits* by Dündar and Unutulmaz of [1].

2.2.2 Electromigration and IR-Drop

AMS ICs suffer from diverse non-idealities that became increasingly more relevant with the reduction of the circuit sizes in the last years, and may cause catastrophic circuit failures. These non-idealities must be taken into account during the circuit design in order to mitigate their effect on the product reliability [56]. Two of these non-idealities are: electromigration, which refers to the material migration in the power networks and signal wires that are stressed with high current-densities, deteriorating the interconnect lifetime; and IR-Drop, that consists of a fluctuation of the net voltage due to the interconnect resistances, affecting circuit behavior and performance [57, 58].

Analyzing the electromigration physical phenomenon, J. R. Black, in 1969, was the first to developed a theoretical model to estimate the median time to failure (MTF) in hours of an interconnect in an IC [59], as presented in Eq. (2.5). The model was developed for the aluminum conductors used during the early years of integration industry.

$$MTF = \frac{A}{J^n} \exp\left(-\frac{\Phi}{k.T}\right) \tag{2.5}$$

where, A is a constant that contains a factor involving the cross-sectional area of the interconnect, J the current-density in amperes per square centimeter, Φ the activation energy in electron volts for the interconnect material, k the Boltzmann constant, T the working temperature of the interconnect, and n is used as a scaling factor. By observing Eq. (2.5) it is notorious that only two parameters can be changed by the designer: the current-density, which can be controlled by designing the proper interconnects for the current imposed on them, i.e., the wider the interconnect is assigned, the smaller is the current-density and subsequently electromigration resistance [58];

and the temperature, which may be indirectly controlled by assigning power and thermally-sensitive devices and interconnects in different areas of the chip, and by considering worst-case conditions in the determination of the current-densities.

While the accuracy of the model of Eq. (2.5) is outdated for today's integration technologies, the principles that the combined effects of current-density and temperature are responsible for the gradual degradation of the interconnect is still valid.

2.2.3 Electromigration-Aware Approaches

Handling currents manually in analog circuits is an iterative and time-consuming process. Electromigration is an emergent topic [60, 61], the works addressing this problem focus essentially on how to deal with the multitude of current-densities observed in AMS circuits, and, how to assign and expand the widths of those interconnects accordingly in the routing step of the layout generation. Targeting to mitigate the negative effects of electromigration on the reliability of electronic interconnects [57]. In the literature, this problem often starts with wire planning, which determines the tree with the flow of currents between terminals (i.e., the set of terminal-to-terminal connections and respective current-flows). These terminals of the same net need to be all interconnected with the minimum wiring area possible (where the wires' width are function of the current imposed on them) and while satisfying the Kirchhoff's current laws in every terminal. And only after this wire planning step, wire routing, that rectilinearizes the required paths, is performed.

The first AMS current-driven routing solutions available in the literature were introduced in [62–64]. There, a semi-automatic 'three-point Steinerization' technique was used to construct a Steiner tree for a set of single-port sources and sinks, with different current-densities, by sequentially adding the nearest terminal to the current sub-tree. Since the algorithm is making a local choice, some obstacles may have to be manually moved after the topology determination for a design rule correct layout.

In [65] a current-driven wiring topology (WT) is automatically extracted by greedily assigning the source-sink edges that maximize the interconnect area gain. An improved version can be found in [66] considering obstacles in the rectilinearization of the paths. An integer linear programming-based algorithm is proposed in [67] to handle multiple currents in the SMT construction, however, the computation time becomes prohibitive as the number of terminals increases. In [68] a WT for analog high current application is obtained by separating the single-port multiterminals into small clusters that are routed independently by an exhaustive procedure, but causing the algorithm to lose the global view of the current-assignment problem. In WiT [69, 70], the original NP-hard wire planning problem is converted to a class P problem via the proof of the greedy-choice property, in this way, the competition among the flow of all sources is considered in the wire planning, unlike all the previous approaches. Finally, in [71] channel space restrictions are considered, in which wider paths avoid narrow channels in the construction of the WT.

2.2.4 Wiring Symmetry

Analog layout parasitics have a huge impact on the on-die circuits' performance, and to match them, the whole design flow is subject to an exhaustive set of stringent constraints. Particularly, symmetry constraints, which are mandatory in the design of differential circuits to guarantee an identical behavior of its two symmetric parts, must be imposed not only to the devices' placement, but also to the interconnects between them, i.e., devices' routing. The objective is to achieve a fully-symmetric wiring in the final layout.

State-of-the-art works on automatic analog layout generation focus on extraction of placement rules along with sophisticated constraint-based floorplan generators [72], but consistently ignoring automatic wiring symmetry. For automatic symmetric routing only a few attempts can be found in the literature. In the early 1990s a set of symmetries between wires were identified manually, and then, the path-finding algorithm defines the paths in the presence of nonsymmetrical obstacles [1]. This can be done by evolving both sides of the layout simultaneously [4], or, by mirroring all the obstacles and then routing only one side of the layout, and consequently mirroring the obtained path to obtain the symmetric one [73]. Only more than two decades later the problem of automatic wiring symmetry was raised again. In [74] pairs of symmetric wires are detected based on a hierarchical constraint tree obtained for the placement, and then, these rules are applied to obtain abstract global paths that can be used to speed up the manual detailed layout generation.

2.2.5 Commercial Solutions

Mentor Graphics' IRoute and ARouter [75] are user-assisted solutions where the designer knowledge is used to manage the routing generation. The designer manually chooses the order in which the nets are routed, and the nets with a higher priority are routed more directly. The wires' width and spacing to other nets must be selected, and also, the designer sets the specific conductor to be used in each net and the transition points between layers. The tool provides markers and directions given the set of actual constraints, to interactively help the designer to manually draw the wires.

Cadence® Virtuoso® Chip Assembly Router [76] is also an interactive tool for transistor, cell, block and chip-level routing. A set of design constraint- and process-rule-driven routing features are provided to support analog designers working on 90 nm design processes or above. The tool performs dynamic real-time checking with enforcement of the design constraints and process rules, supported with push and shove functionalities that avoid the need to move obstructing routing paths. Automatic global and power routing, and detailed completion features, are available to speed-up the manual routing. A real-time connection with Virtuoso® Schematic Editor and Virtuoso® XL Layout Editor allow dynamic cross-probing of instances and nets, with simultaneously editing.

2.3 Complete Layout Generation Tools

In this section, some of the milestones in the analog layout generation, along with some recent tools, are reviewed.

2.3.1 Procedural Generation

In the earliest approaches, procedural module generation techniques coded the entire layout of a circuit in a software tool [77], which would generate the target layout for the parameters attained during sizing. This parametric representation of the layout is fully developed by the designer, either by a procedural language or a GUI. Applied to analog, ALSYN [78] employs fast procedural algorithms that are controlled through a database of structures and attributes. A high-functionality pCell library independent of technologies can be found in [79]. Although fast, these methods lack the flexibility to accommodate wide changes, making the cost of introducing a new design task relatively high and technology migrations may force complete cells redesign.

2.3.2 Template-Based

The use of template approaches, which define the relative position and interconnection of devices, is a common practice. A template-based generation is used by IPRAIL (Intellectual Property Reuse-based Analog IC Layout) [80, 81] to automatically extract the knowledge embedded in an already made layout, and use it for retargeting. Layout retargeting is the process of generating a layout from an existing layout. The main target is to conserve most of the design choices and knowledge of the source design, while: migrating it another given technology; update specifications; or, attempt to optimize the old design [1]. Liu and Zhang [82, 83] developed a tool that automatically conducts performance-constrained parasitic-aware retargeting and optimization of analog layouts. Performance sensitivities with respect to layout parasitics are first determined, and then the algorithm applies a sensitivity model to control parasitic-related layout geometries, by directly constructing a set of performance constraints subject to maximum performance deviation due to parasitics.

In order to retain the knowledge of the designer but without forcing an implicit definition, LAYGEN [54, 84] uses a template-based approach to guide the layout generation. ALADIN [52, 53] also allow designers to integrate their knowledge into the generation process. While ALG [51] uses the same knowledge-based principle, allowing the designer to interact with the tool in different phases but introducing a higher level of abstraction, leaving to the discretion of the designer if the final layout is obtained almost full automatically or by designer directives. In LAYGEN II [85, 86] a deterministic template-based placement over a B*-tree is performed, and

unlike previous approaches, a fully evolutionary approach is used to optimize the routing of any provided terminal-to-terminal connectivity.

In layout description script (LDS) [55, 87, 88] linear programming is used to generate layouts from simple declarative statements. This specific language is intended to code layout templates to be used for layout-aware circuit sizing due to its flexibility and fast generation time. In order to optimize the area of a fixed/template-based non-slicing floorplan, Unutulmaz et al. [89] also formulates the areas of the transistors, capacitors and resistors as convex functions, and then, the floorplan area is minimized by solving a sequence of convex problems. Due to its fast convergence, the approach is intended to be integrated in LDS and find the optimum dimensions of the layout components during the layout-aware circuit sizing.

More recently, Po-Hsun Wu et al. [90] presented a knowledge-based methodology that generates new layouts by integrating existent design expertise. The approach automatically analyzes legacy design data including circuits, layouts, and constraints, and generates multiple layouts for the new design by reutilizing the legacy information as much as possible. In [91], Po-Cheng Pan proposes a mechanism to extract the relationship among wires and design blocks, and that information can then be used for complete layout migration. Not only the behavior of the old designs is preserved, but several metrics for multiple placement generation and wire refinement are included within this process to improve the quality of the resultant layout prototyping.

2.3.3 Optimization-Based

The optimization-based layout generation approaches consist of synthesizing the layout solution using optimization techniques according to some cost functions, with a higher level of abstraction. These tools beyond their flexibility in terms of incremental adding new functionalities, are easier to implement when compared to procedural and template-based [18].

In the area of device-level placement with layout constraints there are some references of special relevance for this review. ILAC [92] uses SA operating over a topological slicing tree, used to limit the search space. However, as overviewed in Sect. 2.1.2, representing the cells by means of absolute coordinates proved to be the most practical solution to implement layout constraints, even though it allows for an infinitely large solution space. This is the approach found in KOAN/ANAGRAM II [4], LAYLA [6], Malavasi et al. [7] and ALDAC [8]. However, these methods are usually slow and not always produce optimal solutions in terms of area and performance. In [93] the sequence-pair representation is used together with a SA-based kernel to perform an analog-based routability-driven adjustment during the placement process. While all these approaches use optimization for placement, the routing algorithms are implementations of the classical deterministic pathfinding algorithms of Sect. 2.2.1.

Table 2.2 Classification of analog tools based on generation techniques

	Procedural	Template	Optimization
Tools	ALSYN [78], Jingnan [79]	IPRAIL [80, 81], Zheng Liu [82, 83], LAYGEN [54, 84], ALADIN [52, 53], LDS [55, 87, 88], Po-Hsun Wu [90], Po-Cheng Pan [91]	ILAC [92], KOAN/ ANAGRAM II [4], LAYLA [6], Malavasi [7], ALDAC [8], Linfu Xiao [93]
		ALG [51], LAYGEN II [85, 86]	
Advantages	(+) Fast processing;	(+) Places modules quickly; (+) Higher abstraction level than procedural; (+) Useful for retargeting for different specifications and technologies.	(+) Higher level of abstraction; (+) Supports wide specifications' change.
Drawbacks	(−) Limited flexibility for specifications' change; (−) Technology migrations force complete cells redesign; (−) High cost of generating the procedural cell.	(−) Still limits the search space; (−) Still do not support wide specifications' change; (−) Designer must add knowledge.	(−) Slow and computationally expensive; (−) Not always optimal solutions in terms of placement area, routing and/or performance.

In Table 2.2, a classification of the analog tools presented in this section based on generation techniques is presented, and a summary of the advantages and drawbacks of each technique is also highlighted. Also, a brief description of the functional specifications of the referred tools is remitted to Table 2.3.

2.3.4 Commercial Solutions

Virtuoso® Layout Suite Family [76] eases the creation and navigation through complex designs, supported by a sturdy multi-window GUI with automatic assistants to aid the designer. These designers' directions guide the physical implementation process while managing multiple levels of design abstractions at device, cell, block, and chip levels, focusing on precision-crafting their designs without sacrificing time to repetitive manual tasks. The suite contains different levels of assistance: basic design-creation and implementation environment; assisted correct-by-construction wire-editing functionalities ensuring real time process design rule correctness; captures and drives common hierarchical design intent from schematic editor; and a set of advanced automated finishing tools to optimize the layout and achieve first time successful silicon.

Table 2.3 Summary of the functional specification of the layout generation tools

	Specifications				
		Placer			
Layout tool	Year	Algorithm	Representation	Router	Coding language
ILAC [92]	1989	SA	Slicing tree	Best-first deterministic maze search	Pascal
KOAN/ANAG II [4]	1991	SA	Absolute	Line expansion with re-routing, over-the-device wiring and crosstalk avoidance	C
ALSYN [78]	1993	Deterministic	Slicing tree	Maze router with crosstalk avoidance	C
LAYLA [6]	1995	SA	Absolute	Minimum spanning tree only	C++
Malavasi [7]	1996			Maze router	OCTTOOLS
Jingnan [79]	2001	Procedural layout generation			SKILL
ALDAC [8]	2002	SA	Absolute	Local routing with two metal layers	C++
IPRAIL [80, 81]	2003–2006	Linear programming and graph-based methods			–
ALADIN [52, 53]	2005–2006	Two-stage technique: (1) GA approach with SA and HPWL estimation; (2) Fast re-annealing placement algorithm and maze global routing			C++, SKILL, Tcl/Tk
LAYGEN [54, 84]	2006–2007	SA	B*-tree	Adapts the template routing to the created floorplan	Java
ALG [51]	2009	Different from custom design to fully automatic mode, combined with global and local routing steps			Java
Linfu Xiao [93]	2010	SA	Sequence-pair	Maze router with two metal layers	C++
Zheng Liu [82, 83]	2010	Mixed-integer nonlinear programming and graph-based methods			C/C++
LAYGEN II [85, 86]	2013	Template-based over B*-tree		Evolutionary approach that optimizes the routing of a provided connectivity	Java
LDS [55, 87, 88]	2011–2014	Linear programming			Java
Po-Hsun Wu [90]	2015	Linear programming over connection graphs representations			C++
Po-Cheng Pan [91]	2015	Linear programming and graph-based methods			C++

Synopsys'® Galaxy Custom Designer LE [49] also offers a set of automation functionalities for layout generation. The automatic guard ring generator creates guard rings in real-time, easily edited by using stretch, reshape and chop options. The connectivity driven capabilities allow to insert vias and arrays of vias by point and click mode or between layers that belong to the same net, this process ensures that all design rules are validated automatically. Furthermore, user-assisted applications like auto connect features that allow to route pins from a higher level, interactive bus routing with different via pattern choices, align assist displays with interactive alignment markers allowing to place layers in a faster manner, commands to specify locations to create bridges or a tunnels for routing, etc. The advantages of these frameworks is that they add several functionalities to speed-up layout design, while maintaining the old features of the classical layout editing environments that most analog designers still rely.

2.4 Closing the Gap Between Electrical and Physical Design

The post-layout performance of a circuit needs to be guaranteed in the presence of layout parasitics, which prevent the circuit from reaching the expected performance values. However, time-consuming and unsystematic iterations between electrical and physical design steps to counterbalance those layout-induced performance degradations, which are illustrated in Fig. 2.5, need to be avoided as much as possible [1]. Furthermore, even the evaluation of the circuit area or aspect ratio is practically impossible from the netlist alone, which is especially critical for designs with geometric constraints, e.g., an upper limit in the layout width, causing expensive redesigns. One possible solution involves the overlap of these two different phases, by including layout-induced effects into the circuit sizing phase, which is overviewed in the next sub-section. Knowing the layout induced effects in the circuit sizing process ensures that the performance of the solution is attained after the layout, and that the geometrical requirements are taken into account in a realistic manner. This methodology can be found in recent literature with different designations like parasitic-aware, layout-aware and layout-driven sizing.

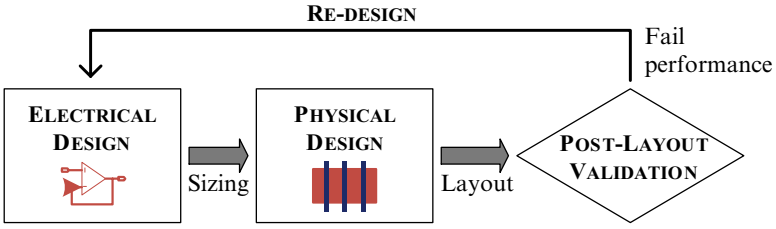


Fig. 2.5 Traditional design flow, iterations between electrical and physical design phases

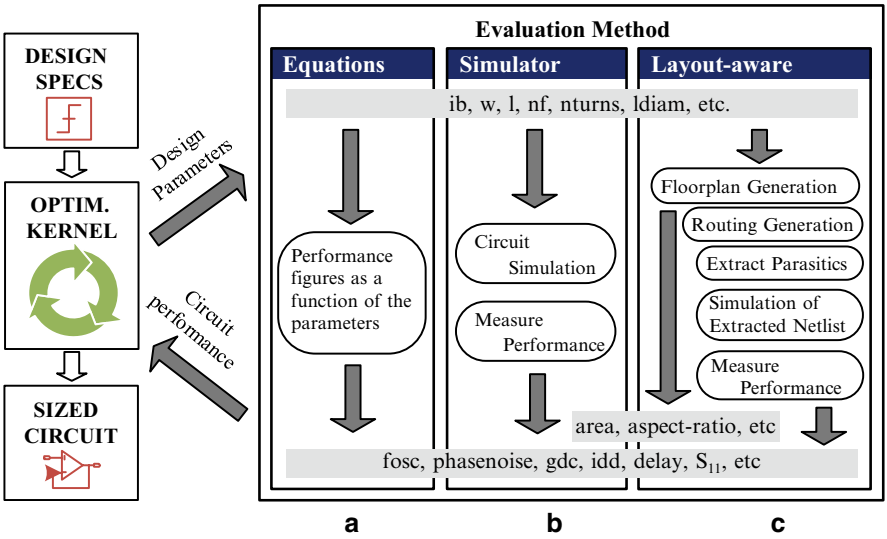


Fig. 2.6 Alternative evaluation methods for optimization-based circuit sizing: (**a**) equation-based, (**b**) simulation-based, and (**c**) layout-aware

2.4.1 Circuit Sizing Task

Before layout, the circuit must be sized. The designer must devise an architecture suitable to implement the system specifications and then all the blocks. From the high-level specifications to the devices sizes all blocks must be properly sized. This task is commonly done manually, the designer starts by finding an approximate solution using simplified analytical expressions, and then iteratively, adjusts the solution until it meets all specifications. Another possibility for circuit sizing is the use of circuits' synthesizers. These tools are used to automate the circuit sizing and can be: (1) equation-based, where the methods use analytic design equations to evaluate the circuit performance or (2) simulation-based, which do not comprise a previous modeling task, but require higher execution time than equation-based due to the verification done using electrical simulations. A general architecture for automated circuit sizing using optimization-based approaches is presented in Fig. 2.6.

2.4.2 Layout Generators Embedded in Layout-Aware Approaches

The inclusion of layout-related data is crucial to achieve a first-time-right design at the end of the optimization loop, trimming down the impact of parasitic devices in analog circuits' performance and providing accurate geometrical layout properties, e.g., area, aspect ratio, etc. To integrate layout generation or layout-related data in

sizing optimization, the traditional flow of optimization-based circuit sizing must be modified according to Fig. 2.6c.

Without actually generating the layout in real-time, in [94] device parasitic effects are modeled by linear regression from a Pareto optimal surface, obtained by sampling the design space and using a procedural generator to produce the layout for each point, where each solution is aware of its specific layout induced effect. In [95] the layout is also produced by a parameterized generator, where layout parasitics and devices sizes are passed to the precompiled symbolic performance model that estimates the circuit performance for each sizing solution, attempting to avoid circuit simulation. In [96] the parasitic effects are modeled in the circuit equations, a template-based layout generator is used to derive the parasitic equations that are solved together with the performance equations using non-linear-programming.

In [97] the procedural generator is used during the execution of a design plan, where part of the parasitic effects are estimated using analytical models, and other part is accounted by setting the BSIM's [98] parameters accordingly. Unlike previous approaches, in [99] the procedural generator is used to generate the layout but inside the optimization loop. Likewise in [100, 101] both parasitic-aware and geometrically constrained sizing are handled. Since the predefined layout template is supported by a slicing tree and the block placement is fixed, area and shape optimization are obtained by finding the number of fingers of MOS transistors that yield desired geometric features. In addition, layout parasitic effects are extracted and used during the circuit's evaluation in the optimization loop. This extraction can be done either using analytical equations and layout sampling or using complete 3D geometric extraction models. To improve the flexibility of the floorplanner, in [102] is applied some effort on minimizing the placement area with convex optimization over the template-based description language LDS [89].

Finally, in [103] the designer knowledge is embedded in a set of placement and routing matching, symmetry and proximity constraints, as well as geometric and electrical performance constraints. Then, a deterministic nonlinear optimization algorithm is used to determine the device sizes. In each iteration, the floorplan is exhaustively explored by enumeration using Plantage [2], and then the best placement, selected based on some performance criteria, is routed and considered for post-layout simulation.

2.4.3 Parasitic Extractors Used in Layout-Aware Approaches

The parasitic structures extracted from the layout must be precise enough to guide the parasitic-aware circuit sizing in the right direction. In [99] a 1/2-D model is chosen but only applied for the area and fringing capacitance of the metal and poly stripes of the circuit's critical nets, which makes the estimation quick, but loses accuracy and needs user intervention to identify the critical nets. This can lead to errors, if, depending on the layout, less critical nets became problematic. In [94, 96] analytical polynomial models with predictor variables, such as diffusion area/perimeter or

even voltages/currents from circuit simulation are used, yielding a very fast estimation of the parasitic impact on circuit's performance without post-layout simulation. Still these values are just predictions and can have a large error. In [97] analytical models are also used, but the predictor variables related to the transistors' stress effects, which are distances between components, do require some post-layout analysis.

In [95, 100–103] an external extractor is used which guarantees accurate results for the parasitic estimation. Although the parasitics obtained by using these techniques are very accurate, their inclusion in the optimization loop either forces the creation of a custom parameterized layout, which adds considerable setup time, or using custom full layout generator, which is prohibitively slow to be inside the loop. Table 2.4 summarizes the overviewed state-of-the-art layout-aware circuit sizing approaches with focus on the layout generator used, and also, geometrical and parasitics included.

2.5 Overview of the State-of-the-Art on Analog Layout Automation

Figure 2.7 establishes a chronological representation of the layout generation tools presented in this Chapter, organized by the generation technique used.

The research goals of the proposed automatic layout generator, AIDA-L, introduced in Chap. 1 of this book are revisited and discussed according to the paradigm found in this state-of-the-art. The driving issues for the development of methodologies described in this book are briefly overviewed as starting point for the next Chapters.

2.5.1 Support User-Assisted Placement Generation

Beyond the efforts made towards the generation of a fully automatic layout capable of competing with expert-made layouts, it is possible to notice that EDA tools are moving in a different direction from two decades ago. There is now a strong attempt to recycle existing layouts, migrating them to new technologies or optimize the old design. Many of the circuits manufactured today are the ones developed and implemented years ago, so it is extremely important to take advantage of the knowledge embedded in their layouts and follow the advances in the integration technologies, instead of going through all the design process again. For this, the idea of parameterized model/template is present in the most recent successful approaches in the past few years. Increasing the designer's active part in the generation of the floorplan is not necessarily a drawback, since the inclusion of his knowledge increases the floorplan quality and allows the tool to easily generate a solution that fully meets analog designers' expectations/needs.

Table 2.4 Comparison between state-of-the-art works on layout-aware sizing with special emphasis to layout-related data included in-loop

	Layout Generator		Layout-related data included						
				Parasitics					
Work	Placer	Router	Geometric	Bulk	Intra-device	Inter-device	Interconnect	Resistances	Observations
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

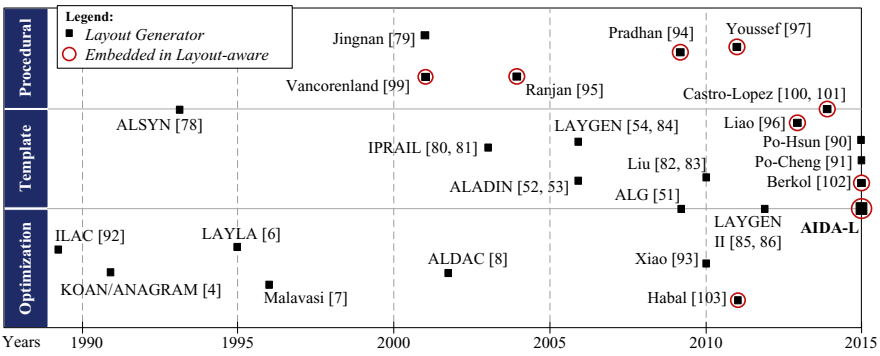


Fig. 2.7 Chronological representation of analog layout generators

Table 2.5 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]	✗	✗	[15–17, 40, 47]
Sequence-pair	[20, 21, 23, 24]	✗	✗	[22]
BSG	[25]	✗	✗	✗
O-tree	[27]	✗	✗	[26]
B*-tree	[28–32, 37–39]	✗	✗	[2]
TCG/TCG-S	[33–35]	✗	✗	✗

2.5.2 Support Fully-Automatic Placement Generation

The main drawback of using topological representations for the exploration of analog placement is the difficulty found on satisfying and maintaining the layout design constraints [18]. While symmetric-feasible conditions were derived for most of the representations, proximity constraints are barely supported and often considered as objective in the weighted single-objective cost functions. Absolute representation is the most intuitive manner of implementing analog constraints, as symmetry and proximity can be forced in any combination, without relying in special symmetric-feasible conditions of the topological structure that encodes it. No packing, structural scan or post-processing time is required to fulfill/implement the analog constraints, the algorithm moves the cells explicitly and implements the requirements inherently.

Topological floorplan representations became a trend on analog design automation in the last two decades. Research community became extremely focused on developing innovative and problem-oriented topological representations and left for background the optimization algorithms used. A summary of the optimization algorithms used in the overviewed tools for analog placement automation is sketched on Table 2.5. The SA-based kernel used nowadays to perturb most of the topological representations is nearly identical from the one used in the disruptive approaches in

the early 1980s. However, the truth is that published absolute representations' floorplans are rather sparse, which further degrade with the scalability of the problem and the results are no match for the most recent topological representations. Nevertheless, recent efforts on stochastic/evolutionary algorithms suggest that searching the extensive search space derived from the use of absolute coordinates can compete with a search within the reduced solution space produced by a topological structure.

2.5.3 *Alleviate Designer from the Routing Task*

From the reviewed approaches, it is possible to note that analog floorplan is a dominating the EDA research community. However, the routing task of the proceeding is where the most of the difficulties remain. This is clear when observing the limitations of the current approaches, and the completely lack of routing automation in commercial EDA, only user-assisted functionalities. Furthermore, maintaining a model/template for routing proved to require huge setup time, and results in a setup that is application dependent, a maintenance effort beyond tolerable.

For the routing approaches considering only wire planning or path assignment abstractly from the silicon level and technology-inherent design rules, simplistic 'dot-models' are used to represent these complex multilayer multiport terminal structures. In these automatic approaches only results for single-net problem solving are presented, which focus on the WT generation, and the applicability to obtain design rule-correct routings for the set of conflicting nets of real analog IC design cases remains to be proved [65–71]. In the traditional automatic approaches for design rule-correct layouts [4, 7, 8, 51, 52, 62–64, 78], besides the traditional deterministic and error-prone implementations, the exact location of the single-port terminals may be either defined in the generation/import of the modules, selected manually or by pin assignment problem solving, which lead to higher setup times and/or sub-optimal solutions.

2.5.4 *Embedding in a Layout-Aware Circuit Sizing Methodology*

Retrieving Table 2.4, in [94, 95, 97, 99] procedural generators, on which the whole layout of the circuit is coded, requiring huge setup time, are used (in-loop or for layout sampling). Also, these generators lack the flexibility to handle change, i.e., changing the topology can discard most or all the previous setup work. In [96, 100–102] the flexibility of the floorplanner is improved by the use of template-based approaches, however, due to the multitude of different sizing solutions found throughout the whole Pareto solution set it is almost impossible to pack all the solutions properly. The alternative is fully automatic generation with an exhaustive search [103], but at expense of an increased computation time.

It is important to notice that for all approaches, the routing setup/template is the same for all design solutions and is not fitted to the solutions as they vary in a multi-

tude of devices' sizes/shapes and performances. However, interconnect parasitic capacitances, pointed as major contributors to performance degradation and signal integrity problems [104], are obviously dependent of the WT, i.e., terminal-to-terminal connectivity, selected and total wiring area and congestion attained by the end of the routing phase. Moreover, routing quality is strongly related with electric-currents measured for the corresponding circuit sizing, but this is seldom considered.

2.6 Conclusions

In this Chapter a set of tools applied to analog IC design automation were presented, with emphasis on the layout generation task. Although much has been accomplished, the fact is that automatic generators are not yet of standard usage in the industrial design environment. The few commercial tools overviewed show that only the user-assisted approaches, that offer control over the generation, found their way into the EDA market. These tools are used to speed up the manual design task by means of interactive functionalities.

Most of the available commercial solutions stand out because of the powerful GUIs provided and their characteristics as layout project managers, but lacking on the algorithmic complexity for the automatic generation. At the same time, the reviewed automation approaches are usually limited to some specific circuit or circuit class, and the large time required to create a new tool or prepare an existing one to support a new circuit, causes analog designers to keep on designing the layout manually.

On the other hand, due to the efforts made in EDA community for enhanced automatic circuit sizing methodologies, layout-aware approaches are spreading and represent an important part of the future of analog design automation. However, closing the gap between electrical and physical design for a unified synthesis process requires efficient and reliable layout generation tools.

References

1. H.E. Graeb (ed.), *Analog Layout Synthesis: A Survey of Topological Approaches* (Springer, New York, 2011)
2. M. Strasser, M. Eick, H.E. Graeb, U. Schlichtmann, F.M. Johannes, Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2008, pp. 306–313
3. D. Jepsen, C. Gelatt, Macro placement by Monte Carlo annealing, in *IEEE International Conference on Computer Design (ICCD)*, 1983, pp. 495–498
4. 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)
5. 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)
6. 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)

7. 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)
8. 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)
9. A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing* (Springer, Berlin, 2003)
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. D.F. Wong, C.L. Liu, A new algorithm for floorplan design, in *Proceedings of the 23th ACM/IEEE Design Automation Conference (DAC)*, June 1986, pp. 101–107
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. Graeb, F. Balasa, R. Castro-Lopez, Y.-W. Chang, F.V. Fernandez, P.-H. Lin, M. Strasser, Analog layout synthesis—recent advances in topological approaches, in *Proceedings on Design, Automation & Test in Europe (DATE)*, Mar 2009, pp. 274–279
19. 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
20. 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)
21. 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)
22. 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)
23. Y.-C. Tam, Y. Young, C. Chu, Analog placement with symmetry and other placement constraints, in *Proceedings EEE/ACM International Conference on Computer-Aided Design*, Nov 2006, pp. 349–354
24. 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
25. 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)
26. P.-N. Guo, C.-K. Cheng, T. Yoshimura, An O-tree representation of nonslicing floorplan and its applications, in *Proceedings of the 36th ACM/IEEE Design Automation Conference (DAC)*, June 1999, pp. 268–273
27. 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

28. 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
29. 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)
30. 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
31. 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
32. 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
33. 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
34. 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)
35. 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
36. 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
37. 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
38. 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
39. 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)
40. 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)
41. T. Chen, T. Kuan, C. Hsieh, C. Peng, Challenges and solutions in modern analog placement, in *International Symposium on VLSI Design, Automation, and Test (VLSI-DAT)*, April 2012, pp. 1–4
42. 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
43. 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
44. 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)
45. 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
46. 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

47. P.-H. Wu, M. Lin, Y.-R. Chen, B.-S. Chou, T.-C. Chen, T.-Y. Ho, B.-D. Liu, Performance-driven analog placement considering monotonic current paths, in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, Nov 2012, pp. 613–619
48. Mentor Graphics' Tanner EDA (2016), Tanner AMS and MEMS Flows. https://www.mentor.com/tannereda. Accessed 30 May 2016
49. Synopsys (2016), Synopsys. http://www.synopsys.com. Accessed 20 May 2016
50. 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
51. Y. Yilmaz, G. Dundar, Analog layout generator for CMOS circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **28**(1), 32–45 (2009)
52. 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)
53. 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
54. 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
55. A. Unutulmaz, G. Dundar, F. Fernandez, A template router, in *20th European Conference on Circuit Theory and Design (ECCTD)*, Aug 2011, pp. 334–337
56. G. Gielen, E. Maricau, P. Wit, Design automation towards reliable analog integrated circuits, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2010, pp. 248–251
57. J. Lienig, Electromigration-aware physical design of integrated circuits, in *18th International Conference on VLSI Design*, Jan 2005, pp. 77–82
58. J. Lienig, Introduction to electromigration-aware physical design, in *Proceedings International Symposium on Physical Design (ISPD)*, Mar 2006, pp. 39–46
59. J.R. Black, Electromigration—a brief survey and some recent results. IEEE Trans. Electron Devices **16**(4), 338–347 (1969)
60. G. Posser, V. Mishra, P. Jain, R. Reis, S. Sapatnekar, Cell-internal electromigration: analysis and pin placement based optimization. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **35**(2), 220–231 (2016)
61. J. Pak, S. Lim, D. Pan, Electromigration-aware routing for 3D ICs with stress-aware EM modeling, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2012, pp. 325–332
62. 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
63. J. Lienig, G. Jerke, T. Adler, Electromigration avoidance in analog circuits: two methodologies for electromigration-driven routing, in *International Conference on VLSI Design Conference*, Jan 2002, pp. 372–378
64. J. Lienig, G. Jerke, Current-driven wire planning for electromigration avoidance in analog circuits, in *Proceedings of the Asia South Pacific Design Automation Conference (ASP-DAC)*, Jan 2003, pp. 783–788
65. 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
66. 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
67. 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)

68. 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
69. 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
70. 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)
71. 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
72. M. Eick, M. Strasser, K. Lu, U. Schlichtmann, H. Graeb, Comprehensive generation of hierarchical placement rules for analog integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(2), 180–193 (2011)
73. E. Malavasi, A. Sangiovanni-Vincentelli, Area routing for analog layout. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **12**(8), 1186–1197 (1993)
74. Y. Yang, I. Jiang, Analog placement and global routing considering wiring symmetry, in *11th International Symposium on Quality Electronic Design (ISQED)*, Mar 2010, pp. 618–623
75. Mentor Graphics (2016) Mentor Graphics. https://www.mentor.com. Accessed: 20 May 2016
76. Cadence Design Systems (2016) Cadence. https://www.cadence.com. Accessed: 20 May 2016
77. R.J. Lipton, S.C. North, J. Valdes, G. Vijayan, R. Sedgewick, ALI: a procedural language to describe VLSI layouts, in *Proceedings of the 19th ACM/IEEE Design Automation Conference (DAC)*, 1982, pp. 467–474
78. V. Meyer, ALSYN: flexible rule-based layout synthesis for analog ICs. IEEE J. Solid State Circuits **28**(3), 261–268 (1993)
79. X. Jingnan, J. Vital, N. Horta, A SKILLTM—based Library for retargetable embedded analog cores, in *Proceedings on Design, Automation and Test in Europe (DATE)*, Mar 2001, pp. 768–769
80. N. Jangkrajarng, S. Bhattacharya, R. Hartono, C. Shi, IPRAIL—intellectual property reuse-based analog IC layout automation. Integr. VLSI J. **36**(4), 237–262 (2003)
81. S. Bhattacharya, N. Jangkrajarng, C. Shi, Multilevel symmetry-constraint generation for retargeting large analog layouts. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(6), 945–960 (2006)
82. L. Zhang, Z. Liu, A performance-constrained template-based layout retargeting algorithm for analog integrated circuits, in *Proceedings of the 47th ACM/IEEE Design Automation Conference (DAC)*, Jan 2010, pp. 293–298
83. Z. Liu, L. Zhang, Performance-constrained template-driven retargeting for analog and RF layouts, in *Proceedings of the 20th symposium on Great lakes symposium on VLSI (GLSVLSI)*, May 2010, pp. 429–434
84. N. Lourenco, N. Horta, LAYGEN—automatic analog ICs layout generator, in *Proceedings of Conference on Telecommunications*, 2007, pp. 165–168
85. R. Martins, N. Lourenço, N. Horta, *LAYGEN II—Automatic Analog ICs Layout Generator Based on a Template Approach* (Genetic and Evolutionary Computation Conference (GECCO), Philadelphia, PA, 2012)
86. R. Martins, N. Lourenço, N. Horta, LAYGEN II—automatic layout generation of analog integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(11), 1641–1654 (2013). doi:10.1109/TCAD.2013.2269050
87. A. Unutulmaz, G. Dündar, F.V. Fernández, LDS—a description script for layout templates, in *European Conference on Circuit Theory and Design (ECCTD)*, Aug 2011, pp. 857–860
88. A. Unutulmaz, G. Dundar, F.V. Fernández, Template coding with LDS and applications of LDS in EDA. Analog Integr. Circ. Sig. Process **78**(1), 137–151 (2014)

89. A. Unutulmaz, G. Dundar, F. Fernandez, Area optimization on fixed analog floorplans using convex area functions, in *Proceedings on Design, Automation & Test in Europe (DATE)*, Mar 2013, pp. 1843–1848
90. 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)
91. P.-C. Pan, C.-Y. Chin, H.-M. Chen, T.-C. Chen, C.-C. Lee, J.-C. Lin, A fast prototyping framework for analog layout migration with planar preservation. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(9), 1373–1386 (2015)
92. J. Rijmenants, J. Litsios, T. Schwarz, M. Degrauwe, ILAC: an automated layout tool for analog CMOS circuits. IEEE J. Solid State Circuits **24**(2), 417–425 (1989)
93. L. Xiao, E. Young, H. Xiaoyoung, K. Pun, Practical placement and routing techniques for analog circuit designs, in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2010, pp. 675–679
94. A. Pradhan, R. Vemuri, Efficient synthesis of a uniformly spread layout aware Pareto surface for analog circuits, in *22nd International Conference on VLSI Design*, Jan 2009, pp. 131–136
95. M. Ranjan, W. Verhaegen, A. Agarwal, H. Sampath, R. Vemuri, G. Gielen, Fast, layout inclusive analog circuit synthesis using pre-compiled parasitic-aware symbolic performance models, in *Proceedings on Design, Automation & Test in Europe (DATE)*, Feb 2004, pp. 604–609
96. Y.-C. Liao, Y.-L. Chen, X.-T. Cai, C.-N. Liu, T.-C. Chen, LASER: layout-aware analog synthesis environment on laker, in *Proceedings of the 23rd ACM International Conference on Great Lakes Symposium on VLSI*, May 2013, pp. 107–112
97. S. Youssef, F. Javid, D. Dupuis, R. Iskander, M-M. Louerat, A python-based layout-aware analog design methodology for nanometric technologies, in *IEEE 6th International Design and Test Workshop (IDT)*, Dec 2011, pp. 62–67
98. N. Paydavosi et al. BSIM4v4.8.0 MOSFET model manual (2013), http://www-device.eecs.berkeley.edu/bsim/Files/BSIM4/BSIM480/BSIM480_Manual.pdf
99. P. Vancorenland, G.V. der Plas, M. Steyaert, G. Gielen, W. Sansen, A layout-aware synthesis methodology for RF circuits, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2001, pp. 358–362
100. R. Castro-Lopez, O. Guerra, E. Roca, F. Fernandez, An integrated layout-synthesis approach for analog ICs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(7), 1179–1189 (2008)
101. A. Toro-Frias, R. Castro-Lopez, F. Fernandez, An automated layout-aware design flow, in *International conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sep 2012, pp. 73–76
102. G. Berkol, A. Unutulmaz, E. Afacan, G. Dundar, F.V. Fernandez, A.E. Pusane, F. Baskaya, A two-step layout-in-the-loop design automation tool, in *IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, June 2015, pp. 1–4
103. H. Habal, H. Graeb, Constraint-based layout-driven sizing of analog circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(8), 1089–1102 (2011)
104. G. Shomalnasab, H. Heys, L. Zhang, Analytic modeling of interconnect capacitance in submicron and nanometer technologies, in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 2553–2556

Chapter 3
AIDA-L: Architecture and Integration

This chapter provides an overview of the proposed design flow for analog integrated circuits (ICs), with emphasis on the automatic layout generation task, using AIDA-L tool. AIDA-L is integrated in an in-house analog IC design automation framework, AIDA (Martins et al., International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Sep 2012, pp. 29–32; Lourenço et al., Design, Automation & Test in Europe Conference (DATE), Mar 2015, pp. 1156–1161; Martins et al., International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Sep 2015, pp. 1–4), for complete automatic design from circuit-level specifications to physical layout description. As such, AIDA-L presents itself in two fronts, first, as a standalone tool that assists the analog designer in process of layout generation; and second, when combined with AIDA-C, in AIDA's framework, it provides complete layout-related data that is included during automatic optimization-based sizing, enabling a layout-aware sizing methodology for analog ICs.

In the first section of this chapter, the capabilities of the tool as a standalone low-level layout generator are sketched. Additional detail about the tool's implementation, inputs, outputs and adopted generation flow is provided in section "Standalone Design Flow", whose interfaces are used by the designer to quickly generate and monitor the automatic generation of the target layout. Finally, in section "Integration on AIDA's Framework", the AIDA-L tool embedded in the AIDA's layout-aware circuit sizing flow is presented.

3.1 Standalone Design Flow

AIDA-L supports the analog designer as a standalone tool in the iterative process of layout generation. There are two levels of abstraction in the tool, *user-assisted* floorplan generation and *fully-automatic* floorplan generation. The *user-assisted* or

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_3

fully-automatic designation is only relative to the detail of the high level guidelines provided by the designer, in both modes all tasks are performed automatically by the tool given the proper set of inputs. As the quality of the routing solution is inherently dependent from the floorplan solution, and due to the discontinuity of previous *user-assisted* in-house routing implementations, only a *fully-automatic* routing is provided in AIDA-L for both floorplan generation modes.

3.1.1 User-Assisted Floorplan Generation

It is acknowledged that layout generation is a knowledge-intensive and contradictory task, where each designer or company has its own layout style. This style is often very regular for a large number of applications, even with some specifications or technological changes, and the design guidelines for the most common cells are kept the same. For simple cells, parametric generators are a valid solution to implement these guidelines, however, parametric generators are specific to a technology making them difficult to reuse. In addition, for cells that are more complex, the development of effective parametric generators has proven ineffective either on design-time or on design-reusability. In order to cope with these issues and increase design efficiency, AIDA-L stores the floorplan regularities (i.e., impositions of the designer) in a layout meta-description that is independent of technology and sizing. The layout design flow using the tool as a low-level generator with *user-assisted* floorplan generation is illustrated in Fig. 3.1.

Assuming that the desired technology design kit is available, given the high level floorplan (commonly designated as template file), devices' sizes, the netlist and an optional set of electric-current values for each terminal (optional, but fundamental to obtain satisfying routing solutions) the target layout can be automatically generated. After the initial definition, the result can be iteratively re-generated by applying adjustments to the topological relations between cells in the template file. This way, the designer can control the generation of the floorplan from a higher level and easily introduce new guidelines, until the desired solution is obtained for the current set of devices' sizes. It is important to note that this template-based generation is a particular case of a *user-assisted* generation, where the user only interacts in the beginning of the process, i.e., setting up the tool, and the whole generation process is accomplished automatically by the tool.

Even though in a manual IC design process the layout generation task is always performed after the specification translation, often designers have to introduce some changes in the project after the layout is concluded. Those changes may occur by different causes, e.g., from a late adjustment to the previous specifications that consequently result in new devices' sizes or even from a sub-block replacement at system-level. In the manual layout design process this redesign may lead to partial or complete loss of the previous work, reflecting in the total time necessary to obtain a satisfying solution. In AIDA-L the information is reused since the guidelines are kept in the template file, making layout adjustment way more efficient.

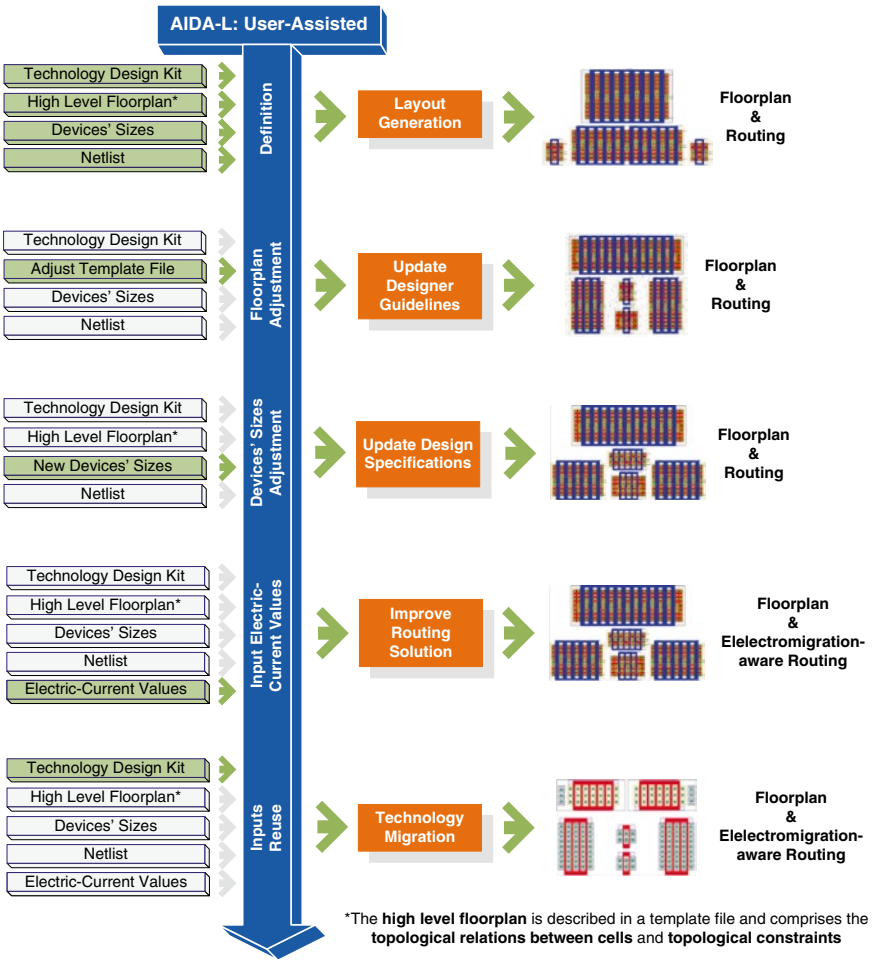


Fig. 3.1 Automatic layout design flow using AIDA-L with user-assisted floorplan generation

The reuse of old designs is a common practice among analog designers. In AIDA-L the same inputs can be used to retarget a circuit for completely different specifications, being at the discretion of the designer if the old template guidelines are still satisfactory for the new design. Although the template file may require some adjustments, these changes in the high level floorplan take an almost negligible time when compared to manual redesign of the layout. Even for technology retargeting, the previous guidelines can be used to regenerate the layout for a different technology.

The netlist is common to all updates or retargeting operations and, a design rule and Layout-versus-schematic (LVS)-clean routing solution is generated directly from the netlist. However, the designer can input the electric-current values for each terminal, which are used as guidelines for the routing process, greatly improving the quality of the routing and enabling AIDA-L to present an electromigration-aware layout solution.

In summary, the inputs can be iteratively changed by the designer, updating high level floorplan guidelines, devices' sizes, electric-current values or choosing new technologies to obtain the desired high quality layout. In this design flow, the analog designer is producing technology- and sizing-independent floorplan descriptions that can be used for retargeting with few or no changes, and the routing is automatically generated from the netlist, making the design flow highly reusable and efficient. This way, designer's efforts and knowledge are better-used due to the increase in design productivity and efficiency, instead of repeating time-consuming tasks.

3.1.2 Fully-Automatic Floorplan Generation

The *user-assisted* mode generates a meaningful and designer-oriented solution from guidelines provided at a high level; still, the tool supports a higher level of abstraction, the *fully-automatic* floorplan generation. However, designers often want a floorplan that has the minimum area possible, that allows the minimum wirelength for the interconnects, and that should also favor proximity between matched devices, space sensitive signal lines from the noisy ones, and so forth. For these reasons it is impossible for a *fully-automatic* tool to present a single best floorplan to the designer. The design flow using the tool as a low-level layout generator with *fully-automatic* floorplan generation is illustrated in Fig. 3.2.

Again, assuming that the desired technology design kit is available, given the topological constraints (which in this work are provided by the designer but can be extracted using a methodology similar to the one proposed in [4]), devices' sizes and the netlist the target layout can be automatically generated. An optional set of current-flows (or signal-flows) and electric-current values for each terminal can be provided, which again, are fundamental to improve the quality of the solutions, both floorplan and routing.

Unlike the *user-assisted* mode that outputs a single solution, the result is a Pareto front of placements representing the tradeoffs between the optimization objectives, if there is any. Although this aspect of the tool is designated by *fully-automatic*, after the generation the designer intervention is of the utmost importance to select the layout solution, from the set of solutions provided, that suits most the current design needs. Each time a new set of devices' sizes is provided, the Pareto front of placements is re-generated.

However, by introducing a desired set of current- and/or signal-flows, these will be reflected in the placement solutions of the Pareto front, improving their quality. Furthermore, the designer can also input the electric-current values for each terminal. Not only the quality of the routing is greatly improved with electromigration-aware solutions, as in the *user-assisted* mode, but also, enables AIDA-L to enhance the automatically generated placements' routability.

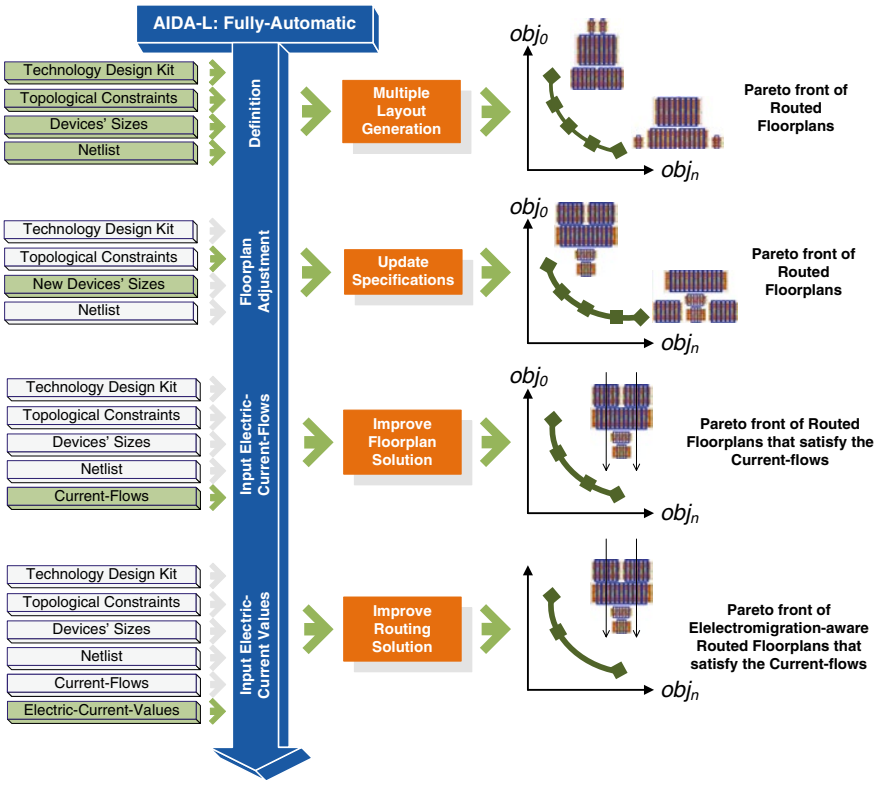


Fig. 3.2 Automatic layout design flow using AIDA-L with fully-automatic floorplan generation

3.2 Standalone Design Flow

The proposed functional architecture is shown in Fig. 3.3 and depicts the principal tasks performed by AIDA-L. AIDA-L is composed of four macro-blocks: Template-based Placer, Optimization-based Placer, Router and Parasitic Extractor.

As overviewed in Sect. 3.1, there are two distinct levels of automation for the placement task: *user-assisted*, which is implemented by the template-based Placer, and *fully-automatic*, powered by the optimization-based Placer. The generation proceeds in the traditional way, first placement and then routing, however, for the optimization-based Placer routing-related data can be included earlier. Lastly, a final validation step must be performed in an industry-standard tool.

In the next sub-sections some details about the input and output parameters, implemented graphical user interface (GUI), technology design kit and adopted automatic layout generation flow are provided.

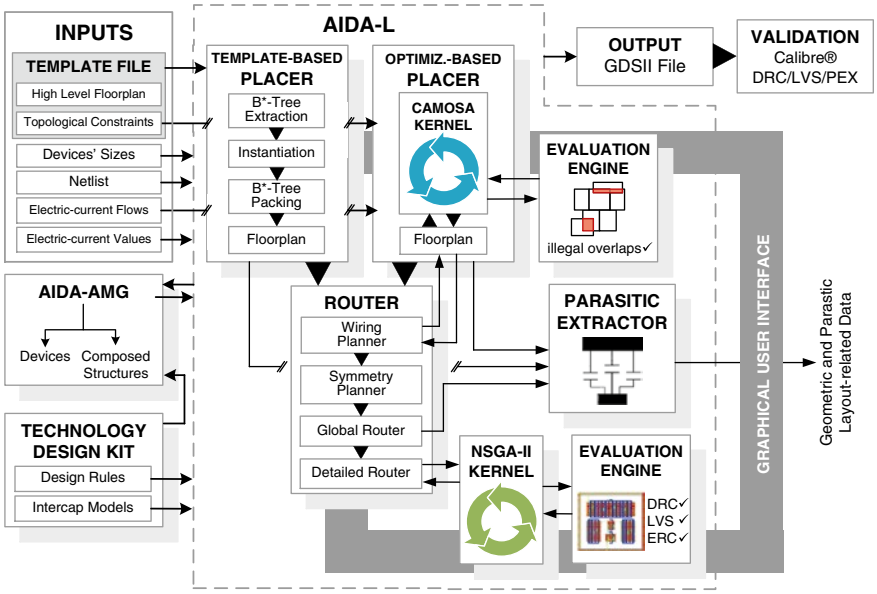


Fig. 3.3 AIDA-L architecture

3.2.1 Inputs

In the next sub-sections the inputs required by AIDA-L, specifically, by the template-based Placer, the optimization-based Placer, the Router and Parasitic Extractor modules are properly detailed.

3.2.1.1 Template-Based Placer: Hierarchical High Level Floorplan and Devices' Sizes

To reduce the unwanted impact of parasitic and process variations, many knowledge-intensive constraints have been considered in analog design. Device matching, symmetry and proximity, thermal effects or current-paths are just some of the factors that analog designers have to consider while planning an analog layout. While designer's expertise is essential in this phase, this knowledge does not require the designer to being aware of the exact details of the technology, e.g., minimum distances or enclosures allowed between layers, etc.

In the template-based Placer, designer's expertise is caught into the technology- and specification-independent template file and used to guide the automatic floorplan generation. The template file contains the high level floorplan, i.e., the topological relations between cells and the topological constraints (a set of symmetry, matching and combine requirements). The template file, the devices' sizes, the AIDA's analog module generator (AIDA-AMG), and the technology's design

rules are used by the tool to automatically deal with the exact placement, and thus, providing a satisfying solution to the designer. The topological relations are used to generate a B*-tree floorplan representation [5], and the extraction and packing procedures using the provided devices' sizes are described together with the template-based placement in Chap. 4 of this book. All this information is described in a human-readable extensible markup language (XML) file [6].

The high level template description can also be used hierarchically as modules, allowing the designer to use template files of simpler cells/blocks in the definition of more complex ones. The use of sub-templates allows to store, in a reusable technology- and specification-independent manner, a set of basic cells that implement specific layout styles for some of the most common blocks. The hierarchy is explored in a bottom-up approach, where the physical representation of the lower levels is generated first, and then, the generation moves up in the hierarchy. It is also possible to perform routing of the lowest levels' templates at a higher level of the hierarchy. Figure 3.4 provides a graphical representation of a hierarchical template definition. In this case, the *M* and *XC* blocks represent devices, and *P* blocks represent sub-templates, containing each of the sub-cells within its boundaries. Each of the represented sub-blocks may have any number of sub-templates, and so on.

3.2.1.2 Optimization-Based Placer: Topological Constraints, Devices' Sizes, Current-Flows and Electric-Current Values

The optimization-based Placer dispenses most of the information contained in the template file for template-based placement, and only requires the topological constraints. Again, the topological constraints, the devices' sizes, the AIDA-AMG, and the target technology's design rules are sufficient for the tool to present multiple placement solutions; however, routing-related data can be included to improve the floorplan quality. If the current-flows and electric-current-values are provided they are reflected in the floorplan solutions, to improve circuit's routability, performance and reliability post-layout.

3.2.1.3 Router: Netlist and Electric-Current Values

For routing the inputs are the floorplan solution generated by the Placer, the netlist, the set of electric-current values for each terminal contained in the netlist and the technology's design rules. The netlist is obviously common to all updates or retargeting operations, as depicted in Figs. 3.1 and 3.2. The set of precise electric-current values (determined in the electrical simulator) are easily obtained and automatically provided to the Router if AIDA-L it is being used integrated in the AIDA framework, as presented in Sect. 3.3. If AIDA-L is being used as a standalone tool, the designer can supply the simulator measure file with the current measures, fill a vector that is provided to the tool with the required electric-current values or leave them empty (i.e., terminals with zero electric-current, which will obviously reflect negatively in the quality of the achieved routing solution).

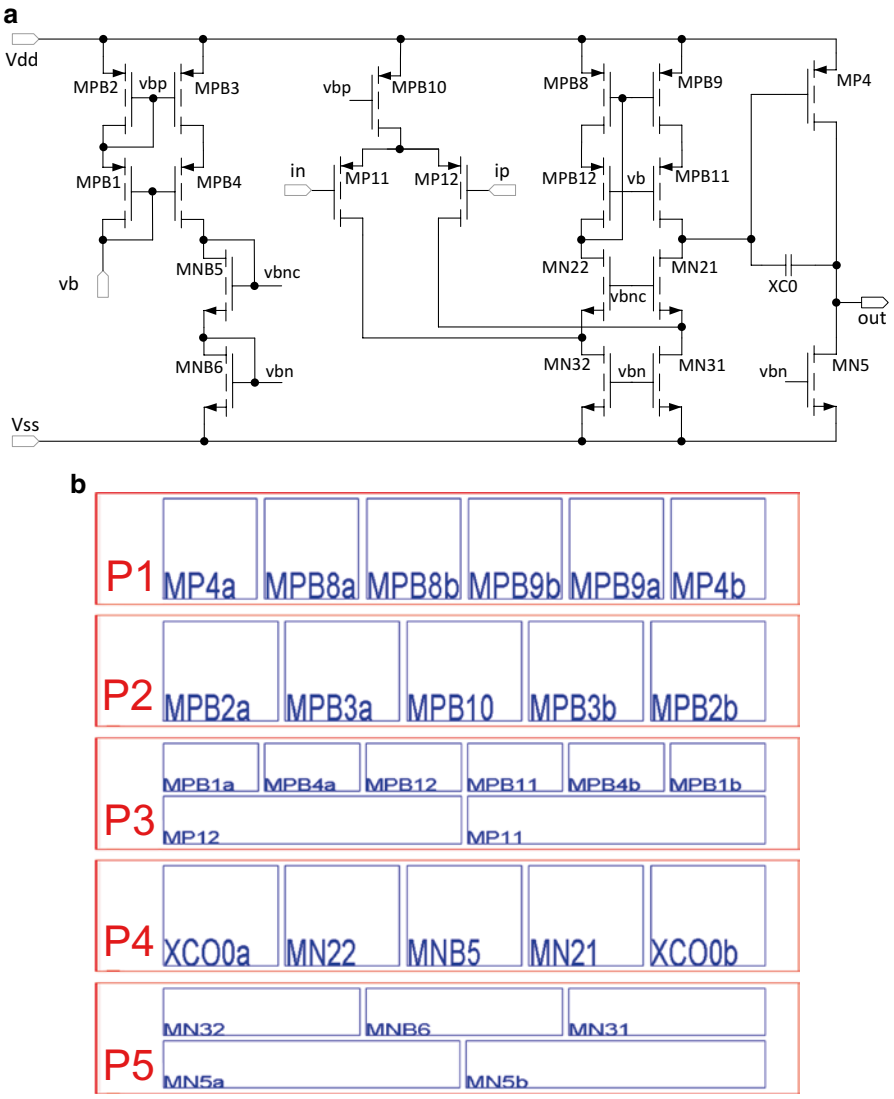


Fig. 3.4 (**a**) Two-stage folded cascode amplifier and the (**b**) Graphical representation of a hierarchical template, P0, P1, P2, P3 and P3 are the sub-templates

For the Parasitic Extractor the inputs are the floorplan and/or a global routing solution, which together with the technology's *intercap* models are sufficient to extract the precise parasitic structures. The parasitic extraction can also be performed over the detailed layout, however, due to the layout-aware methodology proposed in the AIDA's framework, the detailed routing it is not yet considered as input.

3.2.2 Outputs

For layout-aware circuit sizing purposes, the tool can output only the geometric and parasitic layout-related data. However, every layout at any stage of the automatic generation can be saved as a GDSII stream format file, a standard in the microelectronics industry for IC layout data exchange that is supported by the majority of technology vendors and is compatible with nearly all electronic design automation (EDA) software, commercial or free tools. GDSII is a binary file format composed of several structures that are hierarchically related and a set of elements for each structure. The physical validation of the result is, in this work, performed in Mentor Graphics' Calibre® [7] tool, to ensure industrial level design rule check (DRC) and LVS compliance, additionally, post-layout simulations validate the generated design.

3.2.3 Technology Design Kit and AIDA-AMG

In order to define an internal technology design kit the following is required: a layer structure with the enumeration of the layers and vias available; a layer map, containing the GDSII number and type of each layer; the GUI display settings, which describes the colors and patterns used by the layout viewer; a set of design rules that the technology must comply (e.g., minimum distances, encloses or extensions between layers); the *intercap* models, which provide the interconnect capacitive empirical data provided by the foundry; and, finally, the parametric module generator. The United Microelectronics Corporation 130 nm technology design kit has followed the development of the tool, so it is used through this book to generate the examples and test cases.

The AIDA-AMG, which is an internal parametric module generator, is used to instantiate the devices (e.g., MOS transistors, folded and abutted transistor structures, capacitors, etc.) or the simple composed structures (e.g., contact/vias stacks, guard rings, etc.). These parametric modules follow a correct-by-construction generation as they are described in terms of technology design rules (i.e., in a technology-independent manner), which also ease the porting between technologies. Even though there might be large differences between technologies, which are sharper when changing to distant technology nodes, much of the effort taken when defining the parametric modules can be reused when setting up a new design kit. This reuse and adaption of existing module generators may signify only some hours of work, while defining a new module generator from scratch could require some days to perform. At the time of generation or migration, the modules should be properly validated as suited, for a large different number of device sizes.

Each device from the AIDA-AMG is instantiated with multiple electrically-equivalent ports for each terminal, the number of available ports and effective locations are defined in the module generator, and also, the connection between equivalent ports is ensured in the generation of the module. Further detailed about the multiple electrically-equivalent ports is remitted to Chap. 4 of this book.

3.2.4 Graphical User Interface

AIDA-L's framework and GUI are implemented in Java™ 1.7 [8], which is a platform independent programming language. Though efficiency is an important issue, since AIDA-L is under constant development, it is important to keep the modularity and flexibility of the implementation that can be provided by the usage of Java™.

As mentioned, the tool offers a visualization engine and in Fig. 3.5 a screenshot of the GUI is displayed. This interface provides a simple and fast way for the designer to import/save a circuit design, explore the different design solutions, change the optimization parameters, design variables and monitor the evolution of the automatic generation. Moreover, it is also particularly important in the development of the tool, since the impact of introducing new features is easily evaluated in all stages of the generation. The Router internal evaluation procedure automatically places markers in the layout viewer reporting the errors detected, this is essential for the designer to identify the errors, but also for the developer to debug the tool. By contrast, black-box approaches are not suited for multiple reporting facilities, and due to the graphical nature of layout generation it justifies the use of its own interface.

The topological relations between cells contained in the template file are quite suitable for graphical display, in this way, the errors in the template definition are easy to identify in the template viewer. The layout viewer uses the display settings

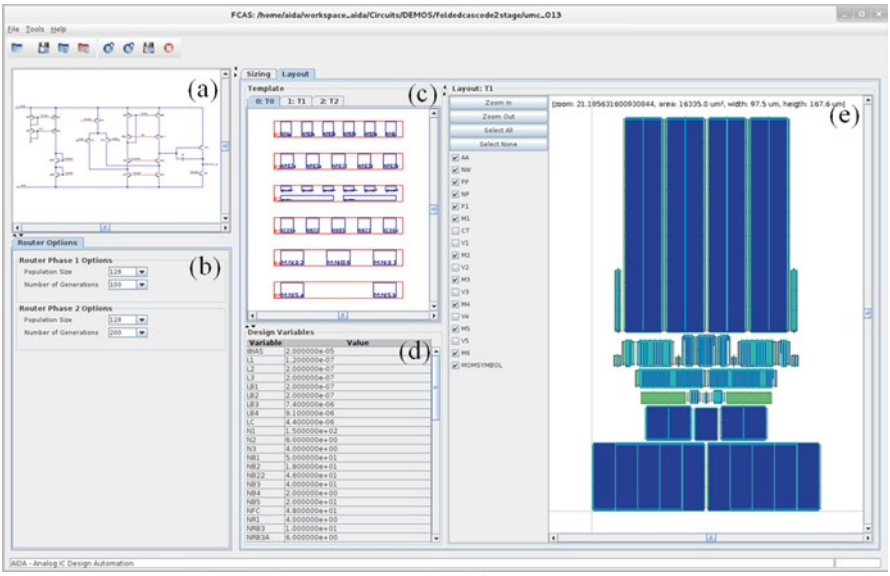


Fig. 3.5 AIDA-L's graphical user interface main panel: (**a**) Circuit illustration; (**b**) Router Options—optimization parameters for the evolutionary detailed routing generation; (**c**) Template Viewer—graphical view of the topological relations between cells contained in the template; (**d**) Design Variables—devices' sizes and constant values for the current circuit design; (**e**) Layout Viewer—the display settings of each layer are used to present the graphical view of the layout

associated to the layout's technology design kit to define the graphical properties, such as color, drawing patterns and z-axis value for each layer. It also implements scroll and zoom functionalities, it provides automatic zoom adjustment to the displaying window when is maximized or resized, and it is also possible to select or unselect a layer for display. The viewer allows the designer to follow the evolution of the layout solution being generated in real time. The GUI can also be used as GDSII viewer, the files can be imported and a library browser is provided to easily explore the layout hierarchy.

There is no need for the designer to use an external layout design tool to accurately visualize the results. Since GUI doesn't allow editing but only visualization, if the designer intends to manually edit some details on the automatically generated layout, the output GDSII file should be imported in a layout editor, e.g., Virtuoso Layout Editor [9].

3.2.5 Automatic Layout Generation Using AIDA-L

In the template-based Placer, the floorplan is obtained by packing the B*-tree representation [5], previously extracted from the high level floorplan contained in the template file, whose implementation is described in detail in Chap. 4 of this book. While in the optimization-based Placer, detailed in Chap. 5 of this book, stochastic computation techniques are used instead to determine the cells' locations. For the latest, an evaluation engine is used to determine the validity of the tentative floorplans.

The Router uses the obtained floorplan as the starting point and then conducts a fully-automatic process. The solution space is then explored ensuring that the routing solution fits in the previously obtained floorplan and the technology design rules are respected, while minimizing the total wiring area and complying with a set of electromigration and IR-Drop constraints. This process is done in three complex deterministic different main phases, denoted as Wiring Planner, Symmetry Planner and Global Router. To finalize the process, the Detailed Router, where multiple sequential executions of the optimization kernel may be used, being that the last execution must result in the detailed and final routing. AIDA-L's Router follows a highly flexible approach, which will be described in detail in Chap. 6 of this book.

Inside the optimization cycle of the detailed Router an internal evaluation procedure is used to evaluate each layout solution. An external validation tool is avoided for several reasons: only a subset of the total technology design rules are required to be validated in the internal procedure, i.e., mostly layers of back end of line; even if the execution times of the external tool can outperform the internal procedures, the communication times can annul that advantage; and also, the parsing of the commercial tool report would mean an extra work when settling a new design kit, deteriorating AIDA-L's modularity, since a new design kit would require a different parser. The design rules are already available for module generation, so they could be equally used in the internal evaluator. This internal procedure provides AIDA-L with the means required to evaluate if a layout will be successfully validated and

thus guide the optimization, also because there is no limiting license fees that would arrive from using an external tool, parallelism can be fully explored.

The Parasitic Extractor uses the floorplan and the global routing solutions to estimate the parasitic resistances of the interconnects, substrate/bulk capacitances, parasitic capacitances between terminals' shapes of the devices, between terminals' shapes and interconnects, and between pairs of global routing interconnects. The module follows an empirical-based approach by processing the *intercap* data provided by the foundry, as described in Chap. 7 of this book.

3.3 Integration on AIDA's Framework

Although the tool can be used as a standalone tool by designers in the layout generation task, is also being developed to be integrated in the bottom-up physical synthesis path of an analog design automation process. The developed tool was integrated on the in-house analog IC design automation environment, AIDA [1–3], which implements a design flow from circuit-level (spice netlist) specifications, with AIDA-C, to a physical layout (GDSII stream) description, with AIDA-L. AIDA results from the integration of these two analog IC design automation tools, AIDA-C and AIDA-L, and the simplified design flow is presented on Fig. 3.6.

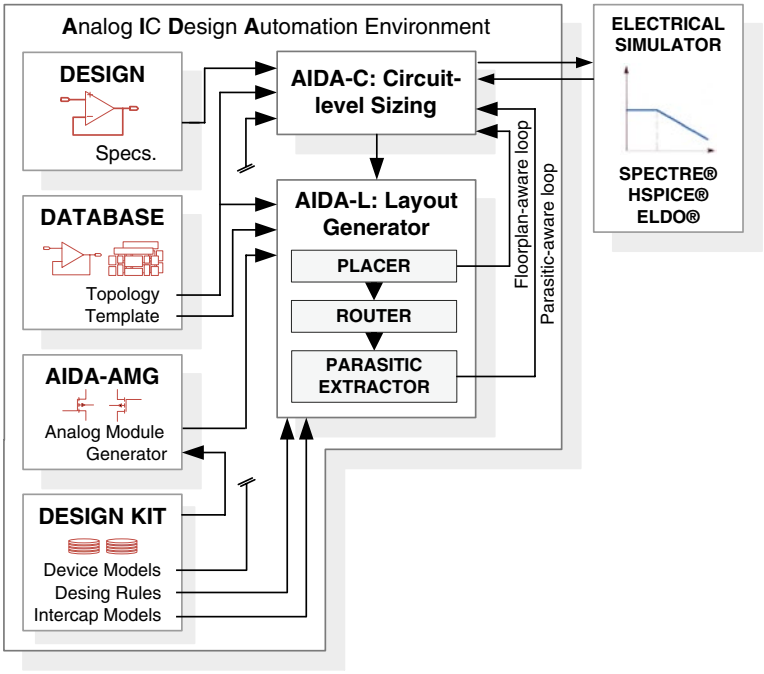


Fig. 3.6 AIDA environment

The automatic circuit-level sizing is carried by AIDA-C [10–13], that accounts for reliability and robustness of the solutions by considering extreme variations, i.e., worst case corner simulations, together with electrical circuit simulators (Spectre® [9], HSPICE® [14] and/or Eldo® [7]) fundamental to evaluate the circuit performance with the high accuracy demanded by the design community. Moreover, the electrical circuit simulator is probably the most well established tool in analog design flow, being used to verify the performance of the circuit since early design stages until post-layout validations. By using the circuit simulator, the inclusion of automation in the design flow, while maintaining accuracy in the obtained solution, is eased.

AIDA-C is based on multi-objective evolutionary optimization kernels and the output, instead of a single solution, is a Pareto optimal front of circuits that fulfill all the constraints and represent the feasible tradeoffs between the different optimization objectives. It is important to note that AIDA-C can optimize several performances figures simultaneously from different testbenches (DC, AC or transient).

After the circuit sizing optimization, the complete layout can be automatically generated for any of the sizing solutions using AIDA-L (also referred as LAYGEN II [15, 16] in the early versions of the framework). Furthermore, due to the integration of AIDA-C with electrical circuit simulators, relevant data can be obtained for each of the different sizing solutions and provided to the layout generator. This is the case of the electric-current/temperature pairs for all the device terminals, which are automatically extracted in this process, and used as guidelines to improve routing quality. AIDA-L's fully automatic Router is able to route any floorplan solution with the same setup. These possible floorplans are different not only in terms of the sizes of the devices, but also, on the modules' relative positioning.

Another important aspect is that AIDA-L interacts with AIDA-C to perform a layout-aware circuit sizing, enabling the inclusion of real geometrical information (e.g., placement's area, width, length, wasted area, etc.) and parasitic structures from the layout during the sizing optimization, *floorplan-aware* and *parasitic-aware* loops of Fig. 3.6, respectively.

3.3.1 Layout-Aware Design Flow

When using AIDA-L to include layout-related data in the layout-aware methodology, is important to guarantee the computational time of the automatic sizing task is not deteriorated. There are two sources of layout-related data to be included during the automatic sizing: first, the geometric information from the floorplan, which can be selected as objective or constraint for optimization; and second, the layout parasitics that are back-annotated in the parameterized netlist and whose impact is reflected in the circuit's performance; i.e., the *floorplan-aware* and the *parasitic-aware* loops, respectively.

The AIDA-L's detailed architecture of Fig. 3.3 is simplified in order to show only the blocks required to include layout-related data during AIDA-C's automatic circuit sizing, and embedded in the complete layout-aware flow with the *floorplan-aware* and *parasitic-aware* loops, as shown in Fig. 3.7.

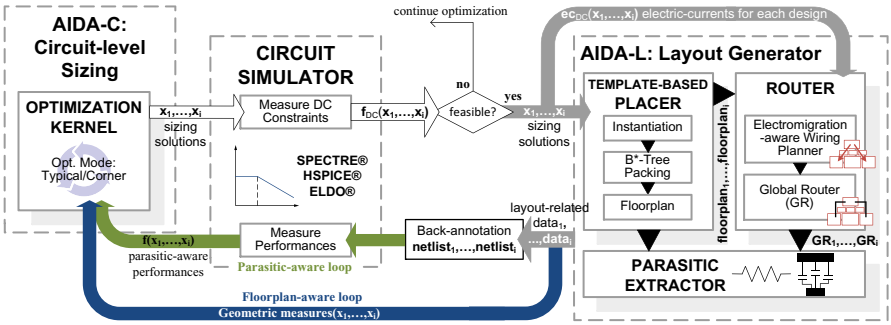


Fig. 3.7 Layout-aware circuit sizing in AIDA's framework, floorplan-aware and parasitic-aware loops

3.3.2 *Floorplan-Aware Loop*

Due to the instantaneous packing of the floorplan from the template file, the template-based Placer is suited to be used inside the automatic circuit sizing optimization, and close the *floorplan-aware* loop to include geometrical information. However, even the flexibility of the non-slicing B*-tree floorplan representation may not be sufficient throughout the whole Pareto solution set of different sizing solutions provided by AIDA-C. The Pareto set encompasses very different circuit solutions and a single template would hardly be the best choice for all, enforcing unreasonable topological constraints in many of them. To prevent such over-constraining of the search space, not one, but multiple topological constraints may be used to allow better packing of the layout throughout the entire solution set. All floorplans are considered and used to produce a placement. After the packing of the floorplan the measures of the best placement are the ones used in the optimizer.

The geometric measures obtained by the template-based Placer can be selected as objectives or constraints, as defined in:

$$
\begin{aligned}
&global\_width, global\_height \\
&area = global\_width \times global\_height \\
&aspect\_ratio = \frac{global\_width}{global\_height} \\
&empty\_area = area - \sum module\_area \\
&empty\_area\_ratio = \frac{empty\_area}{area} \\
&\max\_module\_ratio = \max_{i=1\ D} \left(module\_ratio_i \right) \\
&\quad where\, module\_ratio = \frac{\max\left(module\_width, module\_height \right)}{\min\left(module\_width, module\_height \right)}
\end{aligned}
\tag{3.1}
$$

The *global_width*, *global_height*, *area* and *aspect_ratio* refer to the complete placement, the *empty_area* is the unused space inside the rectangle delimitating the floorplan, the *empty_area_ratio* is the ratio between the *empty_area* and the *area*, and the *max_module_ratio* measures the maximum aspect ratio of the modules and is usually used to constrain the shape of the modules, usually used as functional constraint to keep individual devices approximately square.

3.3.3 Parasitic-Aware Loop

While the floorplan-aware accounts for geometric properties explicitly, the parasitic effects are handled implicitly by a careful layout and aiming at simulating using models as accurate as possible. The advantage of the *floorplan-aware* loop is the reduced impact in the execution time, but for hard specifications the results can still be inaccurate. As overviewed in the state-of-the-art, for the cases where parasitic effects cannot be alleviated by imposing only topological constraints, the *parasitic-aware* loop was added, where the complete layout is generated (placement and routing) and a detailed extraction is done. However, automatic layout generation that is required for accurate parasitic extraction is the bottleneck of layout-aware methodologies. Generators integrated in earlier approaches rely on parametric cells, which lack of flexibility and require huge setup time before each design. The approaches that avoid parametric generators, still, do not update the routing accordingly as the solutions found vary in a multitude of devices' sizes/shapes and performances, and, show extremely high execution time.

AIDA-L's Parasitic Extractor accurately computes the impact of parasitic devices in performance degradation from both floorplan and early-stages of routing. The AIDA-L's Wiring Planner and Global Router generate in the loop of the automatic sizing the layout for each tentative solution at each generation. The computational efficiency is improved by avoiding the need of a time-consuming detailed layout generation needed by off-the-shelf tools for parasitic extraction. Moreover, prior knowledge of the circuit's parasitics is not required, i.e., there is no need for circuit specific equations or models.

By following the optimization process presented on Fig. 3.7, and after the introduction of the *floorplan-aware* and *parasitic-aware* loops, i.e., complete layout-aware loops, the major steps performed are highlighted by order:

1. AIDA-C selects i different sizing solutions, each one with a new set of x design variables (e.g., devices' widths, lengths, number of fingers, etc.);
2. For each sizing solution i the DC constraints are measured using the electrical simulator, also, the DC electric-currents for each terminal are obtained (in order to be provided to AIDA-L). If a solution is unfeasible (i.e., a DC constraint violated) the layout-related data is not considered for further optimization, otherwise, the solution is provided to AIDA-L;

3. AIDA-L's template-based Placer generates *k* floorplans (i.e., the *k* different topological constraints) for each sizing *i*, and the one that suits most the geometrical requirements is provided to the AIDA-L's Router;
4. In the AIDA-L's Router, for each sizing *i*, the floorplan and the set of electric-currents for each terminal obtained with the previous DC simulation, are used to devise an electromigration-aware wiring topology and global routing, properly detailed in Chap. 6 of this book;
5. The parasitics are extracted using the AIDA-L's Parasitic Extractor, which is detailed in Chap. 7 of this book, and back annotated in the *i* different netlists;
6. The parasitic-aware performances are measured from the complete set of testbenches (DC, AC, TRAN, etc.) using the electrical simulator for all defined process, voltage and temperature (PVT) corners and used together with the accurate geometrical properties of the circuit in the optimization process.

As referred, the output of AIDA-C is a family of Pareto non-dominated sized circuits. However, when using the layout-aware flow, the non-dominated sized circuits in the Pareto front meet all the specifications in the presence of layout parasitics and geometrical requirements, representing feasible tradeoffs between the different optimization objectives.

3.4 Conclusion

Analog IC layout design is complex and yields long development time. In this chapter the AIDA-L tool architecture, that aims to reduce the layout design time by the introduction of a complete methodology with several layers of automation, was presented. The proposed methodology allows the designer to control the floorplan generation at a higher level, leaving AIDA-L responsible for dealing with the exact placement and fully-automatic routing, while attending the specific design rules of the target technology and the device sizes specific to the target application. AIDA-L focuses on the efficiency of retargeting operations, introducing new guidelines, update specifications or migration of designs to different technologies are now tasks easier to perform in a project supported by the proposed tool. Moreover, in the absence of the designers' guidelines a fully automatic layout generation process is supported, where in addition to the fully-automatic routing, the tool is also capable of fully-automatic floorplan exploration.

This chapter also introduced two new methodologies to include layout effects in the layout-aware sizing optimization loop. In the *floorplan-aware* loop only geometric information is considered explicitly, with negligible impact in the performance due to the fast generation of the template-based Placer. Then, a *parasitic-aware* loop, where the inclusion of parasitic structures is done using the floorplan and global routing. In the latter, the efficiency on the estimation of the parasitic effect is increased by the AIDA-L's Parasitic Extractor executed over a partial layout solution, instead of using the traditional extraction over a completed and detailed layout. The detailed description of the techniques used for computing placement and routing is found in the next chapters.

References

1. R. Martins, N. Lourenço, S. Rodrigues, J. Guilherme, N. Horta, AIDA: automated analog IC design flow from circuit level to layout, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sep 2012, pp. 29–32
2. N. Lourenço, R. Martins, N. Horta, Layout-aware synthesis of analog ICs using floorplan & routing estimates for parasitic extraction, in *Design, Automation & Test in Europe Conference (DATE)*, Mar 2015, pp. 1156–1161
3. R. Martins, N. Lourenço, A. Canelas, R. Póvoa, N. Horta, AIDA: Robust layout-aware synthesis of analog ICs including sizing and layout, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sep 2015, pp. 1–4
4. M. Eick, M. Strasser, K. Lu, U. Schlichtmann, H. Graeb, Comprehensive generation of hierarchical placement rules for analog integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **30**(2), 180–193 (2011)
5. 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
6. O'Reilly xml.com (2016), eXtensible Markup Language. http://www.xml.com. Accessed: 20 May 2016
7. Mentor Graphics (2016), Mentor Graphics. https://www.mentor.com. Accessed: 20 May 2016
8. Oracle Java (2016), Java. http://www.java.com. Accessed: 20 May 2016
9. Cadence Design Systems (2016), Cadence. https://www.cadence.com. Accessed: 20 May 2016
10. M. Barros, J. Guilherme, N. Horta, in *Studies in Computational Intelligence*. Analog Circuits and Systems Optimization Based on Evolutionary Computation Techniques, vol 294, (Springer, Heidelberg, 2010)
11. M. Barros, J. Guilherme, N. Horta, Analog circuits optimization based on evolutionary computation techniques. Integr. VLSI J. **43**(1), 136–155 (2009)
12. N. Lourenço, N. Horta, GENOM-POF: multi-objective evolutionary synthesis of analog ICs with corners validation, in *Proceedings of the Fourteenth International Conference on Genetic and Evolutionary Computation Conference (GECCO)*, July 2012, pp. 1119–1126
13. N. Lourenço, A. Canelas, R. Póvoa, R. Martins, N. Horta, Floorplan-aware analog IC sizing and optimization based on topological constraints. Integr. VLSI J. **48**, 183–197 (2015)
14. Synopsys (2016), Synopsys http://www.synopsys.com. Accessed: 20 May 2016
15. R. Martins, N. Lourenço, N. Horta, LAYGEN II—automatic analog ICs layout generator based on a template approach, in *Genetic and Evolutionary Computation Conference (GECCO)*, Philadelphia, PA, July 2012
16. R. Martins, N. Lourenço, N. Horta, LAYGEN II—automatic layout generation of analog integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(11), 1641–1654 (2013). doi:10.1109/TCAD.2013.2269050

Chapter 4
Template-Based Placer

This chapter covers AIDA-L's template-based Placer. In short, this Placer first extracts the topological relations described in the template file to a non-slicing B*-tree layout representation (Chang et al., *Proceedings of the 37th ACM/IEEE Design Automation Conference (DAC)*, 2000, pp. 458–463). Then, the B*-tree is packed using the modules, which are instantiated from the AIDA's analog module generator (AIDA-AMG) (Canelas et al., *Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design*, IGI Global, Hershey, PA, 2014), to generate the floorplan that respects the high level floorplan guidelines provided by the designer.

The first section of this chapter covers the overall architecture of the template-based Placer, followed, by the description of the high level floorplan guidelines contained in the template file in section "XML Description for Template-Based Placement". The B*-tree layout representation and extraction procedure are detailed in section "B*-Tree Extraction", and, in section "Instantiation: AIDA's Analog Module Generator", a brief overview of the instantiation of the modules and the characteristics of the parametric module generator are provided. In section "B*-Tree Packing", the floorplan packing from the B*-tree layout representation is overviewed, followed, in section "Case Study: Simple Differential Amplifier", by a simple case study.

4.1 Template-Based Placer Architecture

The proposed architecture/design flow is shown in Fig. 4.1, which depicts the main tasks performed by the template-based Placer: B*-tree Extraction, Instantiation and B*-tree Packing [1, 2].

The usage of a template file, properly detailed in sect. 4.2, that incorporates expert knowledge about the high level floorplan allows AIDA-L to present a meaningful and designer-oriented solution. The Placer acts as a macro-cell placer without

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_4

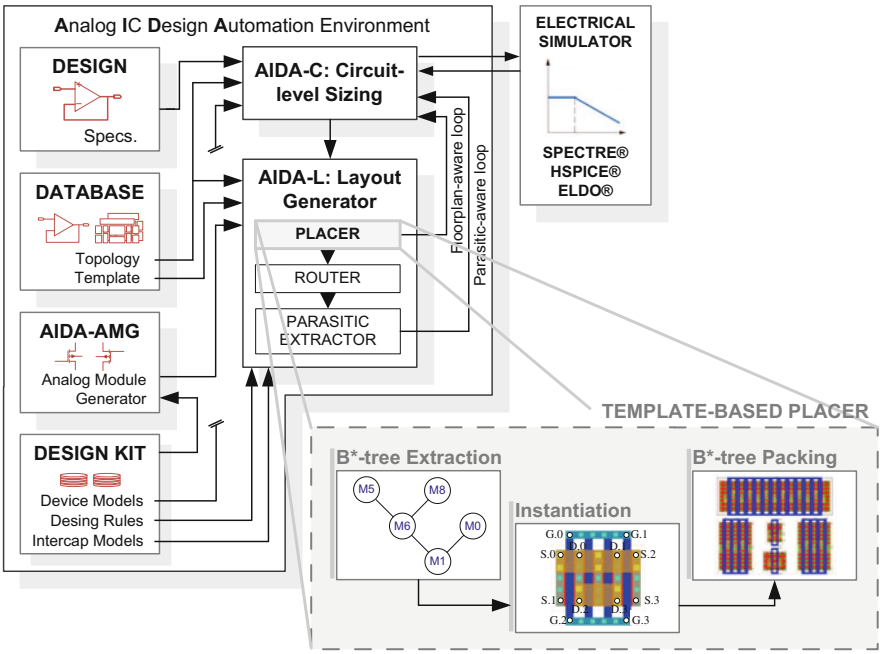


Fig. 4.1 Template-based Placer architecture: B*-tree Extraction, Instantiation and B*-tree Packing blocks embedded in the AIDA-L's design flow

overlap, and a cell can be as simple a transistor and as complex a system-level cell. In addition, since technological details are treated automatically by AIDA-L's Placer, the designer's efforts are focused on difficult layout issues and on the high level floorplan guidelines, and not on the technological dimensions details. This way, designer's efforts are better-used, increasing design productivity, efficiency and design reusability.

The Placer's design flow follows three simple ideas. In the first block, the B*-tree Extraction (Sect. 4.3), the template file, which is semi-automatically generated by the tool from the netlist, and then completed by the designer is processed to extract the B*-tree layout representation [1] that encodes the specified floorplan constraints.

In the second block, Instantiation (Sect. 4.4), all modules contained in the template file are replaced by their physical layout representation. The modules can be instantiated from the AIDA-AMG, sub-templates generated recursively, or custom hand-made layouts imported in GDSII format. Combine operations between modules (described in Sect. 4.4.4) and biasing considerations are taken into account in this phase.

The last block is the B*-tree Packing (Sect. 4.5), here the Placer uses the sizes of the devices and packs the B*-tree using the $O(n.log(n))$ packing algorithm presented in [3] to obtain a compact floorplan.

4.2 XML Description for Template-Based Placement

The information contained in the XML description is the type and relative placement of the cells, and the symmetry, matching and combine requirements. The high level floorplan of each cell is described by a box shape, the sizes of this box have no meaning, and only the relative positioning between cells (boxes) is of concern. Symmetry is specified by two properties, symmetry group that defines a set of cells that share the same symmetry axe; and symmetric cell, the cell that is placed symmetrically in relation to the symmetry axe. Matching can be enforced in the parameterized sizes of the devices to define a set of cells that the designer deemed suited to be matched, proximity is inferred from location of the cells in the template. Finally, combine operations can be used to replace a group of basic cells for more complex layout structures, if available, e.g., merged structures or interdigitized/common-centroid layout styles.

4.2.1 Automatic Generation from the Netlist

With all the information present in the template the initial coding would be time consuming, to alleviate the designer task the template file is semi-automatically generated from the circuit netlist. The parameterized netlist for the simple differential amplifier [4] of Fig. 4.2, where matching constraints are already forced in the devices sizes (width, length and number of fingers) for the transistor pairs: MP_3/MP_4, MN_1/MN_2 and MN_0/MN_b, is illustrated in Fig. 4.3. The template description of Fig. 4.4 was automatically generated by the tool from the information contained in the parameterized netlist.

This automatic process is performed only once for each circuit and all subsequent changes in the designer guidelines result only from small and intuitive adjustments in the template file, as presented next.

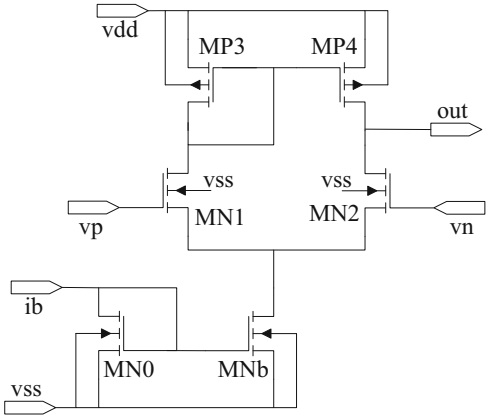


Fig. 4.2 Simple differential amplifier schematic

```
.subckt diff_amp vp vn out ib vdd vss

MP3   n0    n0    vdd   vdd   P_12_HSL130E   W=_w3   L=_l3   M=_nf3
MP4   out   n0    vdd   vdd   P_12_HSL130E   W=_w3   L=_l3   M=_nf3
MN1   n0    vp    n1    vss   N_12_HSL130E   W=_w1   L=_l1   M=_nf1
MN2   out   vn    n1    vss   N_12_HSL130E   W=_w1   L=_l1   M=_nf1
MN0   n1    ib    vss   vss   N_12_HSL130E   W=_wb   L=_lb   M=_nfb
MNb   ib    ib    vss   vss   N_12_HSL130E   W=_wb   L=_lb   M=_nfb

.ends
```

Fig. 4.3 Parameterized netlist of the simple differential amplifier

```
1.  <?xml version="1.0" encoding="ISO-8859-1"?>
2.  <!DOCTYPE Template SYSTEM "template4.dtd">
3.  <Template name="TMP_NAME">
4.     <CellList>
5.        <Cell name="MP3" symGroupId="-1" symCellId="-1" rotate="RCCLK_0">
6.           <Box x="0000" y="0000" w="1000" h="1000" />
7.           <MOSFET type="P" width="_W3" length="_L3" nf="_NF3"/>
8.        </Cell>
9.        <Cell name="MP4" symGroupId="-1" symCellId="-1" rotate="RCCLK_0">
10.          <Box x="0000" y="0000" w="1000" h="1000" />
11.          <MOSFET type="P" width="_W3" length="_L3" nf="_NF3"/>
12.       </Cell>
13.       <Cell name="MN1" symGroupId="-1" symCellId="-1" rotate="RCCLK_0">
14.          <Box x="0000" y="0000" w="1000" h="1000" />
15.          <MOSFET type="N"  width="_W1" length="_L1" nf="_NF1"/>
16.       </Cell>
17.       <Cell name="MN2" symGroupId="-1" symCellId="-1" rotate="RCCLK_0">
18.          <Box x="0000" y="0000" w="1000" h="1000" />
19.          <MOSFET type="N"  width="_W1" length="_L1" nf="_NF1"/>
20.       </Cell>
21.       <Cell name="MN0" symGroupId="-1" symCellId="-1" rotate="RCCLK_0">
22.          <Box x="0000" y="0000" w="1000" h="1000" />
23.          <MOSFET type="N"  width="_WB" length="_LB" nf="_NFB"/>
24.       </Cell>
25.       <Cell name="MNB" symGroupId="-1" symCellId="-1" rotate="RCCLK_0">
26.          <Box x="0000" y="0000" w="1000" h="1000" />
27.          <MOSFET type="N"  width="_WB" length="_LB" nf="_NFB"/>
28.       </Cell>
29.    </CellList>
30. </Template>
```

Fig. 4.4 XML template file automatically generated from the netlist of Fig. 4.3

4.2.2 Designer Guidelines

Starting from the automatically generated template file presented in Fig. 4.4, the designer may now provide the customized high level floorplan guidelines by editing different fields in the XML file.

- First, the relative positioning of each cell in the 2D floorplan space, i.e., the *x* and *y* values in *Box* field of each cell (*w* and *h* fields are optional, and should be considered primarily for those cells whose dimensions are expected to differ considerably from the remaining). For the provided template this can be done by editing lines 6, 10, 14, 18, 22 and 26 of the file; for example the cells' positioning can be set as presented in Fig. 4.5:
- The next step is to specify the symmetry properties of each cell, symmetry group (symGroupId) and a symmetric cell (symCellId). Where '−1' in the symmetry group field is set by default to define a cell without symmetry requirements (symGroupId="−1"), otherwise, if a symmetry group is assigned, '−1' in symmetric cell field is used to define a self-symmetric cell (symCellId="−1"). For the provided template this can be done by editing lines 5, 9, 13, 17, 21 and 25 of the file; for example the cells' symmetry properties can be set as presented in Fig. 4.6:
- Rotate and flip operators are also available to customize the floorplan, however, the cells are all set by default in the same orientation (rotate="RCCLK_0").

Sub-templates can also be defined as suited to ease the design and maintenance of complex circuits. In the current example, due to its simplicity, only one template file that contains all cells is used. After all the proposed editions the resulting and complete template file is presented in Fig. 4.7a.

The template view for the topological relations between cells is shown in Fig. 4.7b. Although these intuitive guidelines can be easily introduced in the XML file, this required setup could greatly benefit if the full potentialities of the graphical user interface (GUI) are explored, e.g., if the designer could move the template cells explicitly by using drag-and-drop functionalities. These features are kept in mind for future revisions of the GUI.

Additional guidelines are the combine operations, which are used to increase the floorplan quality using specific layout structures. Combine operations can be defined to replace two or more basic cells for complex layout structures, if available in the module generator. These structures may be merged structures of transistors, interdigitized or common-centroid layout styles, or any structure available in the parametric module generator and intended to be used by the designer. This can be done by adding new *combine* lines in the end of the XML file. For example, some combine operations could be set as presented in Fig. 4.8, where operation type *CC* stands for common-centeroid, *ID* for interdigitized, and *AB* for abutment (or merging).

Minimum spacing technology design rules are obviously ensured between modules in the packing operation, however, the minimum distances between cells may prove insufficient to perform the routing, and the designer may want to force some gaps between cells for routing. In AIDA-L's template-based Placer, this is possible by using the routing channel construct. The routing channel is treated during place-

a

```
6.  <Box x="0000" y="3000" w="1000" h="1000" /> </-- Cell MP3 -->
(…)
10. <Box x="1500" y="3000" w="1000" h="1000" /> </-- Cell MP4 -->
(…)
14. <Box x="0000" y="1500" w="1000" h="1000" /> </-- Cell MN1 -->
(…)
18. <Box x="1500" y="1500" w="1000" h="1000" /> </-- Cell MN2 -->
(…)
22. <Box x="0000" y="0000" w="1000" h="1000" /> </-- Cell MN0 -->
(…)
26. <Box x="1500" y="0000" w="1000" h="1000" /> </-- Cell MNB -->
```

b

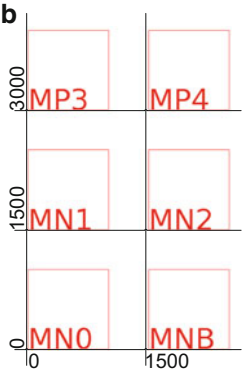


Fig. 4.5 (**a**) Editing the XML template file from Fig. 4.4: Relative positioning of each cell in the space; (**b**) Graphical representation of the relative positioning between cells

```
5.  <Cell name="MP3" symGroupId="1" symCellId="1" rotate="RCCLK_0">
(…)
9.  <Cell name="MP4" symGroupId="1" symCellId="1" rotate="RCCLK_0">
(…)
13. <Cell name="MN1" symGroupId="1" symCellId="2" rotate="RCCLK_0">
(…)
17. <Cell name="MN2" symGroupId="1" symCellId="2" rotate="RCCLK_0">
(…)
21. <Cell name="MN0" symGroupId="1" symCellId="3" rotate="RCCLK_0">
(…)
25. <Cell name="MNB" symGroupId="1" symCellId="3" rotate="RCCLK_0">
```

Fig. 4.6 Editing the XML template file from Fig. 4.4: Symmetry information and rotate operators

ment as any other cell, except that its layout representation is an empty box. This allows the designer to specify areas intended for routing, which eliminates the need to consider wiring congestion during the packing operation.

When all the small adjustments of the template file are concluded, the high level floorplan is mapped to the non-slicing B*-tree layout representation, as described in the next section.

a

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Template SYSTEM "template4.dtd">
<Template name="TMP_NAME">
    <CellList>
        <Cell name="MP3" symGroupId="1" symCellId="1" rotate="RCCLK_0">
            <Box x="0000" y="3000" w="1000" h="1000" />
            <MOSFET type="P" width="_W3" length="_L3" nf="_NF3"/>
        </Cell>
        <Cell name="MP4" symGroupId="1" symCellId="1" rotate="RCCLK_0">
            <Box x="1500" y="3000" w="1000" h="1000" />
            <MOSFET type="P" width="_W3" length="_L3" nf="_NF3"/>
        </Cell>
        <Cell name="MN1" symGroupId="1" symCellId="2" rotate="RCCLK_0">
            <Box x="0000" y="1500" w="1000" h="1000" />
            <MOSFET type="N"  width="_W1" length="_L1" nf="_NF1"/>
        </Cell>
        <Cell name="MN2" symGroupId="1" symCellId="2" rotate="RCCLK_0">
            <Box x="1500" y="1500" w="1000" h="1000" />
            <MOSFET type="N"  width="_W1" length="_L1" nf="_NF1"/>
        </Cell>
        <Cell name="MN0" symGroupId="1" symCellId="3" rotate="RCCLK_0">
            <Box x="0000" y="0000" w="1000" h="1000" />
            <MOSFET type="N"  width="_WB" length="_LB" nf="_NFB"/>
        </Cell>
        <Cell name="MNB" symGroupId="1" symCellId="3" rotate="RCCLK_0">
            <Box x="1500" y="0000" w="1000" h="1000" />
            <MOSFET type="N"  width="_WB" length="_LB" nf="_NFB"/>
        </Cell>
    </CellList>
</Template>
```

b

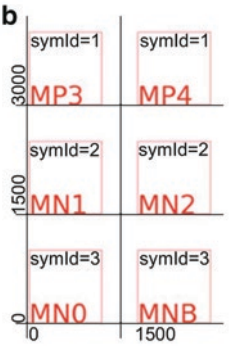


Fig. 4.7 (**a**) XML template file after the designer guidelines; (**b**) Graphical representation of the topological relations between cells described in the template file (*template 1*)

```
<Combine operation="ID" symGroupId="1" symCellId="-1">
   <CellRef id="MP3" />
   <CellRef id="MP4" />
</Combine>

<Combine operation="ID" symGroupId="1" symCellId="-1">
   <CellRef id="MN1" />
   <CellRef id="MN2" />
</Combine>
```

Fig. 4.8 Editing the XML template file from Fig. 4.4: Combine operations

4.3 B*-Tree Extraction

The B*-tree representation imposes vertical and horizontal positioning constraints: each device in the left sub-tree is above its parent device; and if the y-projections of the two devices are overlapping, the device of the node visited first in a pre-order transversal of the tree (visit any node before its left and right sub-trees) is to the left of the device whose node is visited second [1]. In the template the relative positioning of each cell is described by a box shape (x position, y position, width and height), only the relative positioning and sizes between boxes is of concern.

The B*-tree extraction procedure is done using the Algorithm 4.1 [9]. First, all cells are placed in a list ordered by y-position and ties solved with x-position, the bottom-left-most cell is added recursively to the root of the B*-tree being built and removed from the list, this procedure is repeated until all cells were added to the B*-tree. When the y-projections of the cell being inserted and the tree node are overlapping, if the cell is to the right of the node, the cell is added to the right sub-tree, if the cell is to the left of the node's cell, the cell replaces the node's cell, and the node's cell is added to its right sub-tree. When the cell is above the node's cell, there are two scenarios. The cell is above with some x-projection overlapping, in this case the cell is placed in the left sub-tree. Alternatively, the cell is to the right of the current node, in this case there are two possible B*-tree encodings. The tree is copied, and the cell is placed in the left sub-tree in one copy and in the right sub-tree in the other. The complexity of this procedure is due to the fact that it does not use the center of mass of the boxes, but it takes into account the overlaps in X and Y during the extraction of the B*-tree. The overlaps are considered to limit the B*-tree alternatives extracted to the ones that pack in a similar manner to the template, for devices with approximately the same relative sizes.

Algorithm 4.1: B*-tree extraction procedure

input: **List<Cell<**x_c, y_c, $width_c$, $height_c$**>>** *C* (list of template cell boxes)
output: **List<B*-tree>** *B* (a set of B*-tree that code the placement described in the template)

```
1.  List SC:= sort C using BottomLeft comparator
2.  List B:= new list with one empty B*-tree
3.  for each Cell c in SC do // add the cell c to the B*-trees in B
4.     for each B*-tree b in B do
5.        List newB:= new empty B*-tree list
6.        b.root:= addCell(b.root, c) // function addCell(TreeNode, Cell) described below
7.     B:= B + newB // add the newly created trees to B before proceeding to the next cell
8.  return
9.  function addCell(TreeNode node, Cell c)
10.    if node is null then
11.       return new TreeNode linked to c
12.    else
13.       {isAbove, isBellow, isRight, isAboveRightSubtree} = getTopologicalRelations(c, node.c)
14.       if isAbove && isAboveRightSubtree && isRight then // create a copy and add to the
       rigth
15.          copy := copy of b
16.          newB := newB + copy // add tree to the list of newly created trees
17.          copyNode := node from copy with cell equals to node.c
18.          copyNode.right := addCell(copyNode.right,c)
19.          node.left := addCell(node.left, c) // and add to the left of the original tree
20.       else if isAbove && isRight then // not above the right sub-tree
21.          node.right:= addCell(node.right, c) // don't create another B*-Tree
22.       else if isAbove then // and overlap in X
23.          node.left := addCell(node.left, c)
24.       else if isBellow then // this only happens when a node is passed its sub-tree
25.          node.left := addCell(node.left, node.c)
26.          node.c:= c
27.       else if isRight then node.right:= add(node.right, c)
28.       else if not isRight then //and overlap in Y
29.          node.right:= addCell(node.right, node.c)
30.          node.c:= c
31.    return node
32. end function
```

To demonstrate the B*-tree extraction and since the template example of Fig. 4.7 is too simple, the extraction procedure is applied instead to the generic sub-template P1 of Fig. 3.4 of this book, where different B*-tree encodings were obtained. Some steps of the extraction procedure and a set of the final encodings (three of the eight possible final B*-tree) are presented in Fig. 4.9.

4.4 Instantiation: AIDA's Analog Module Generator

In the instantiation step, all modules contained in the template are replaced by their physical layout representation. The modules can be internal procedural generators from the AIDA-AMG, sub-templates that will be generated during the instantiation

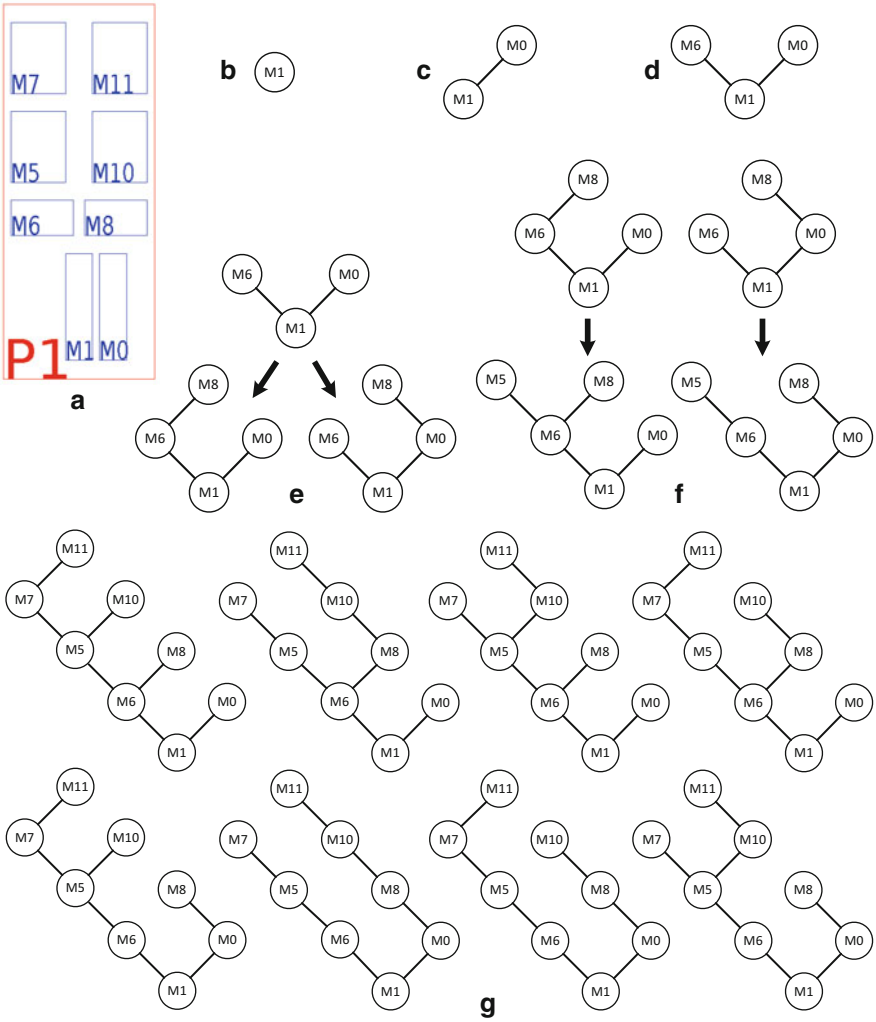


Fig. 4.9 B*-tree encodings extracted using Algorithm 4.1 on the (**a**) sub-template P1: (**b**) Adding M1; (**c**) Adding M0; (**d**) Adding M6; (**e**) Since M8 can be added in the *right* sub-tree of M6 or in the *left* sub-tree of M0, a new tree is created; (**f**) Adding M5; (**g**) Eight possible final B*-tree extracted

phase of the main generation or custom hand-made layouts loaded into the database in GDSII format. Special handling is required when designing custom cells to be used in AIDA-L, since it is necessary to identify the terminals and pins of the cell in a manner consistent with the tool to be used by the Router. Nevertheless, the use of an industry standard allows the use of previous designs as modules

whenever suitable, and there is no limit on each cell complexity, a cell can be as simple as transistors and as complex as amplifiers, because they are used as macro-cells by the Placer.

4.4.1 Supported Structures

The need to efficiently create complex layout structures for the devices is what gave rise to parametric module generators. These tools should be able to design customized simple and complex structures for a device or group of devices, like folded transistors, common-centroid layout styles of transistors or capacitors, merged structures, etc. [5]. The use of these tools helps reducing the errors and alleviates the design time in the analog layout generation task, as most used cells are instantiated by the parametric generators and not constantly custom designed by hand whenever a new set of device's sizes arrives. Also, it allows to take a bottom-up approach in the layout design flow, where the devices are already created in symmetric structures (including intra-cell routing) that facilitates the next stage, the placement process, and results in a more global symmetric distribution of all the devices in the final layout. The actual AMG integrated with AIDA framework is capable of creating device structures using both simple and complex layout styles, following the principles presented in the chapter "Enhancing an Automatic Analog IC Design Flow by Using a Technology-Independent Module Generator" of [6].

Several structures available in the AIDA-AMG are listed. The basic folded transistors, where all fingers are located in the same stack of active area as illustrated in Fig. 4.10a, and also, stacked rows of transistors controller by the parameter *nrows* that defines the number of rows to be drawn, Fig. 4.10b. By using multiple rows, larger devices can be split into smaller ones in different rows, which adds more control over the device aspect ratio while maintaining the original *W*, *L*, and *nf*. Merged transistors stacks, which are created with the abutment of a terminal of two or more transistors, presented in Fig. 4.10c. Interdigitated and common centroid transistors-pairs, on which the first improves matching by interleaving the fingers of two transistors, while the second does the same by arranging the devices in such a manner that they share a common center of mass, depicted in Fig. 4.10d and Fig. Fig. 4.10e, respectively. The increase of matching achieved with the use of these structures reduces the gap between pre- and post-layout simulation.

Furthermore, two intra-device transistor routing processes are available, the first, shown in Fig. 4.10a, connects the common terminals over the device using mostly *metal2* layer, while the second, shown in Fig. 4.10b–d, connects the common terminals outside the device using the *metal1* layer. In the AIDA-AMG the devices are already created in symmetric structures (including intra-cell routing) that facilitate a bottom-up approach in the layout design flow, resulting in a

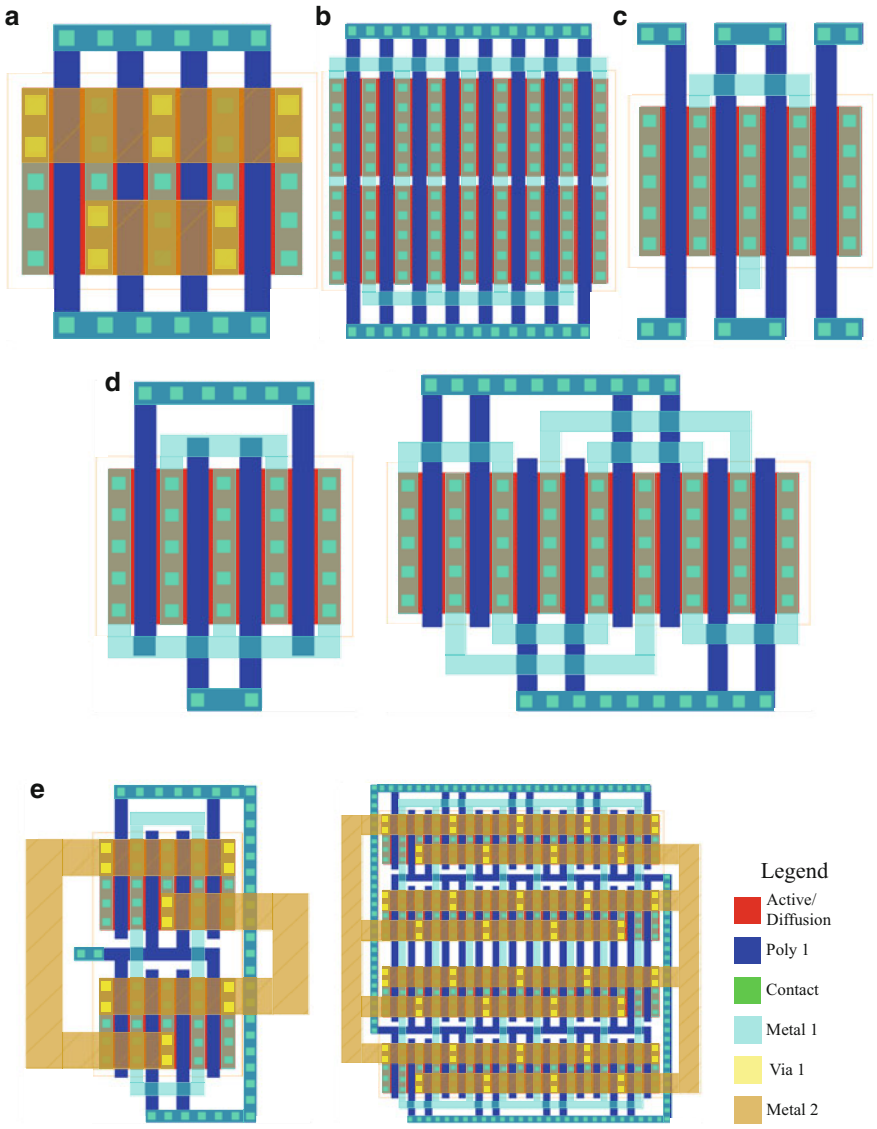


Fig. 4.10 Example of transistors produced by the AIDA-AMG: (**a**) Folded transistor of four fingers with connections over the device; (**b**) Folded transistor with 16 fingers, two rows and connections outside the device; (**c**) Merge of three transistors; (**d**) Two examples of interdigitated of two transistors, each pair with two and four fingers, respectively; and (**e**) Two examples of common-centroid of two transistors, each pair with four and 32 fingers respectively [10]

placement that is globally and locally symmetric. The passive devices supported are Metal-Oxide-Metal capacitors, Metal-Insulator-Metal capacitors and poly-silicon resistors.

4.4.2 Biasing

In a layout design, it must be ensured that the biasing is as close as possible to the active devices. The noisy signals affecting the substrate or the well are sunk by the biasing and scarcely affect the circuit itself. Typically, any unused silicon space should be used for biasing purposes [7]. From the viewpoint of the automatic generation, it is necessary to ensure that there is enough space to add biasing. If these considerations are done after the modules' placement in some cases there might not be enough space to place biasing close enough to the active devices, besides the reduction in performance, this may even breach design rules. The solution used by AIDA-L is to include biasing directly on the generation of the modules, if this functionality is available on the module generator. If it is not available, the designer should efficiently use p-type and n-type guard rings that can be instantiated with AIDA-L composed structures.

The Placer analyzes the netlist (source and bulk terminals) of each device, and then, the suited modules are replaced by an equivalent architecture with biasing considerations. In Fig. 4.11 is provided an example of NMOS transistors with substrate biasing. The modules that have the bulk at equal potential of source or drain terminal allow the use of a topology of Fig. 4.11a, these modules are essential for routing because there is no need to add bulk terminals in the device, it is only necessary to keep the connection to the device source. In case the net assigned to the bulk is not the same net assigned to the device's terminal, it uses the topology of Fig. 4.11b. For the latest, the netlist must be updated to include the new bulk terminals.

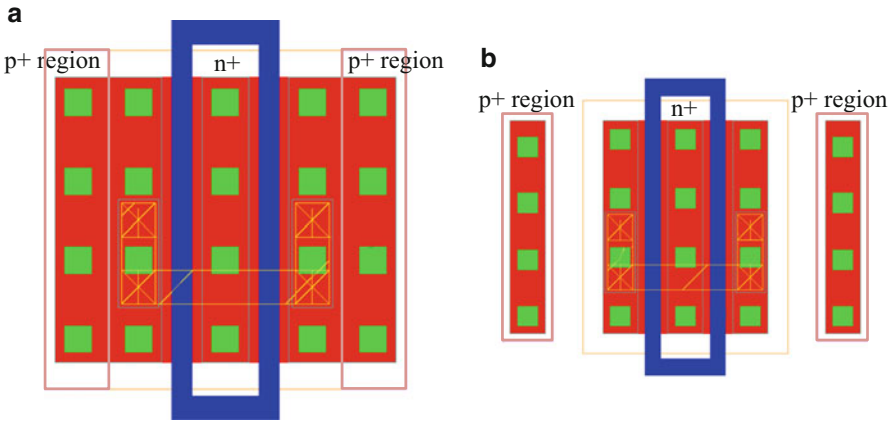


Fig. 4.11 AIDA-AMG: Transistors with well/substrate contacts: (**a**) NMOS with substrate biasing merged; (**b**) NMOS with substrate biasing separated

4.4.3 Handling of Complex Layout Structures

Matched devices appear separate from one another in schematics and template file but they can be combined in the layout using the *combine* operator, as presented before. These complex layout structures for matched devices depend on the structures available on the parametric module generator, e.g., interdigitized layout styles, that interleave two transistors among each other, or, common-centeroid layout styles, to create a circular symmetry of two transistors. Also, merged or abutted structures create an overlapped connection between two cells without introducing design rule violations or connectivity errors, which not only saves space and reduces the wiring length, but in some cases also improves performance by decreasing parasitics. In the case of abutted structures the two cells do not need to be mandatorily matched, e.g., different total widths but with the same width per finger.

However, these parametric cells have some options, e.g., which is the terminal overlapping (source or drain) and if the gates are connected. The Placer analyzes the netlist provided and verifies if the devices are connected in the same net, either by source or by the drain, and also, analyses the gates. If the two share the same source, a common-source topology is adopted, otherwise, a common-drain topology is used. If the gate connectivity is verified, the module is generated already featuring connection between the two transistors' gates. This feature is important to automatically create pins between the two sides of the complex layout structures, preserving symmetry. If no terminals are shared, the cells are kept the same, and no *combine* changes are performed. Since some terminals are merged in the process and cells' names changed, at this stage it is necessary to update the netlist to ensure coherence after the replacement of modules.

Besides saving space, complex layout structures also reduce the length of the interconnect wiring, since in this process some wires disappear and thereby obtaining a simpler final layout. Since routing is the final task of the proposed flow, the quality of the routing is strongly affected by the floorplan generation step [8].

4.4.4 Multiport Terminals

In a cell, the ports of a terminal are identified with a label. The parametric module generators should provide that same label placed over the shapes that are part of the terminal. In order to clarify, henceforward, when referring to the terminals of a cell, e.g., a transistor source, the term terminal is used. When referring the possible contact points of a terminal, e.g., the exact location over the stripe of metal where the connection can be made, the term used is port. The next paragraphs provide more detail of the elements used to describe the routing template. During the instantiation of the modules, after the rotations (if required), the topological labels are assigned to the pins. These labels provide the absolute location and layer information for the pins composing that terminal in the module being generated.

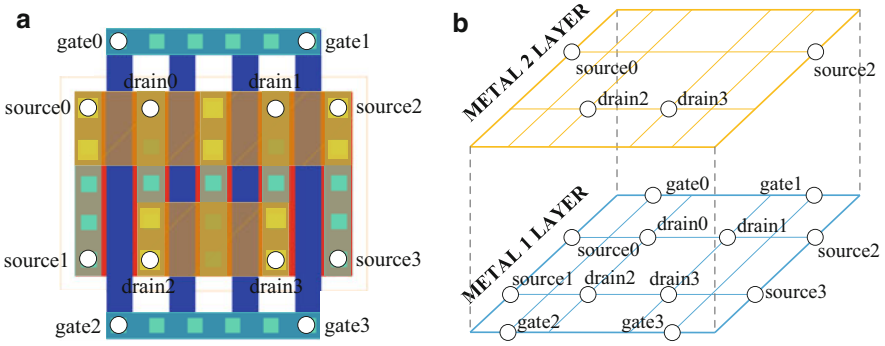


Fig. 4.12 AIDA-AMG: (**a**) Folded transistor with multiple electrically-equivalent ports for: Source, drain and gate terminals. (**b**) Multilayer transversal view of the terminals ports'

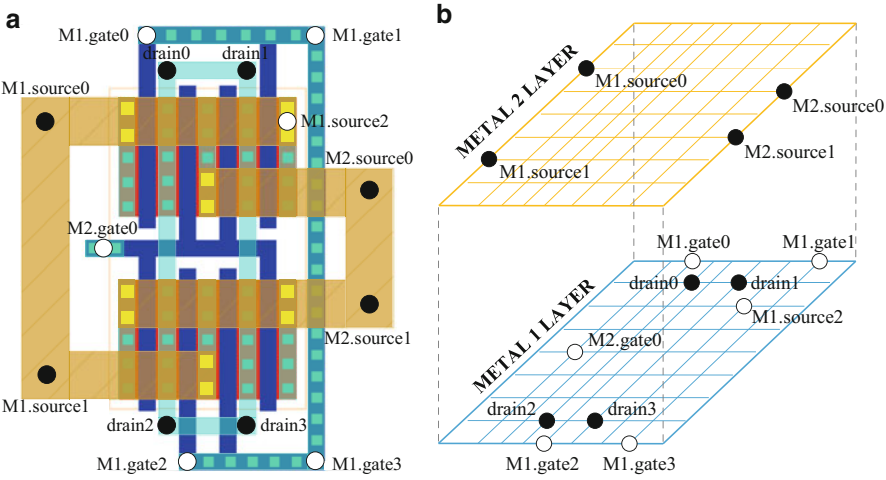


Fig. 4.13 AIDA-AMG: (**a**) Two folded transistors, M1 and M2, in a common-centeroid layout style with multiple electrically-equivalent ports for: Source, gate and common drain terminals. (**b**) Transversal view of the terminals ports' location, for the two layers used

In Fig. 4.12 a simple folded CMOS transistor is presented showing the multiple ports for each terminal. While in Fig. 4.13 an example of two folded CMOS transistors, M1 and M2, in a common-centroid layout style are illustrated, where each terminal has also multiple ports that should to be considered to increase the routing efficiency.

A more complicated terminal geometry than the ones presented can easily have tens of ports located on multiple fabrication layers. The use of the multiport approach increases the flexibility of the devices' placement, allowing cell to be placed in different orientation, while simultaneously enhances the feasibility and quality of the wiring symmetry.

In all the reviewed routing approaches in Chap. 2 of this book, the terminals of the signal nets have only one port that can be routed, i.e., single-port multiterminal nets, which simplify the problem. While single-port cells may be suitable in digital or at system-levels of analog and mixed-signal circuits, where the terminals of the macro-cells are usually located at higher layers of the manufacturing process, at analog cell-level they degrade the quality of the solutions, since most of the device's ports are located in lower layers.

4.5 B*-Tree Packing

All extracted B*-tree are considered and used to produce a placement. To generate the target placement, the packing is performed in two steps, the y-coordinate of the cells are calculated in one pre-order transversal of the B*-tree, where each cell positioning is set by knowing the position of its left parent. Then, the x-coordinate of each cell is computed using the Red-Black interval tree algorithm [3]. With the y-coordinates already assigned, the modules are placed in the smallest available x-coordinate that do not yield any overlap. Since multiple different B*-trees may be previously extracted, each one must be packed for the current devices' sizes and the one that represents the smallest empty area is selected. Whenever the guidelines are changed (not to confuse with devices' sizes), a new set of B*-tree must be extracted before performing a new packing.

During the floorplan packing the complete device is not really needed, as the algorithm only needs to know the space that the device occupies. Hence, a simplified method that generates only the bounds for the devices is provided by the AMG. By using this method instead of the complete detailed device generation, the process is further accelerated.

The best placement is selected as the one with the smaller empty area, from those who are feasible, i.e., meet all constraints (geometrical constraints are only applied in the case when the template-based Placer is used in the *floorplan-aware* loop, as introduced in Chap. 3 of this book). If no feasible solutions are found, the most feasible is selected, i.e., the one where the sum of violated constraints is larger (closer to zero). If two solutions are equally infeasible, the one with less empty-area value is selected. The complete packing procedure is detailed in Algorithm 4.2.

4.6 Case Study: Simple Differential Amplifier

Retrieving the simple amplifier introduced in Figs. 4.2 and 4.3, schematic and parameterized netlist, respectively, in this Section the previously described steps are now applied to real devices' sizes. In Table 4.1 three different design solutions for the simple differential amplifier for the United Microelectronics Corporation (UMC) 130 nm design process are presented.

Algorithm 4.2: Multi-B*-tree packing procedure [10]

input: **List**<B*-Tree> *B* (the set of B*-tree already extracted)
List<Constraint> *C* (the constraints over floorplan measures, e.g., width < 20μm)
Solution *s* (the sizes of the devices)
output: Floorplan *bestF* (the floorplan yielding the smaller empty-area while satisfying the constraints)
Measures *m* (the measures associated with the best floorplan)

```
1.  bestF := null; m := null
2.  for each B*-tree b in B do
3.     try
4.        (meas, placement) := pack(b, s)
5.     catch AMGCannotInstantiateModuleException
6.        meas := INVALID_MEAS
7.     if ((bestF is null) or (isBetter(meas, m, C))) then
8.        bestF:= placement; m := meas
9.  return (bestF, m)
10. function pack(B*-tree b, Solution s)
11.    lib := {} //empty map
12.    sub-placement:= [] //empty list
13.    // generate the sub floorplan recursively
14.    for each B*-tree sb in b.sub-floorplans do
15.       placement := pack(sb, s)
16.       sub-placement += placement
17.       lib += {sb.id: placement.bounding-box}
18.    // instantiate the modules (bounding-box only), if the combination of
19.    // parameters is invalid the Analog Module Generator raises the
20.    // AMGCannotInstantiateModuleException
21.    for each Module m in b.modules do
22.       bbox := AMG.instantiateBox(m, s)
23.       lib += {m.id: bbox}
24.    // pack the B*-tree using the Red-Black interval tree algorithm [3].
25.    // for the boxes in the lib, it returns the bounding box of the layout and
26.    // a map with the location of each module/sub-floorplan in the placement
27.    (bounding-box, device-placement) := rbBTreePack(b, lib)
28.    // compute the measures associated with the resultant packing
29.    meas := computeMeasures(bounding-box, lib, sub-placement) area, width, length,
30.    ratio…
31.    return (meas, device-placement)
32. end function
```

4.6.1 Floorplan Generation: Design 1

The designer guidelines contained in the template file of Fig. 4.7, henceforward called *template 1*, were developed for design 1. All the devices are generated from the AIDA-AMG for the current UMC 130 nm design kit. The resulting floorplan after the B*-tree extraction, instantiation and B*-tree packing is presented in Fig. 4.14. In Fig. 4.14a only basic cells were considered to pack the floorplan, while in Fig. 4.14b the combine operations specified in Fig. 4.8 are included (interdigized layout styles for the transistor pairs MP_3/MP_4 and MN_1/MN_2).

Table 4.1 Three different designs for the simple differential amplifier of Fig. 4.2

		Design 1	Design 2	Design 3
Area (mm^2)		2.123e−4	4.5903e−4	6.060e−4
gdc (dB)		41.585	45.845	47.049
Phase margin (°)		84.981	75.335	77.678
gbw (MHz)		10.08	10.10	10.01
MP_3 MP_4	w3 (µm)	5.25	10.50	9.75
	l3 (µm)	1.90	3.45	3.00
	n3	2	2	2
MN_1 MN_2	w1 (µm)	9.75	20.5	23.5
	l1 (µm)	1.65	3.10	4.35
	n1	2	2	2
MN_0 MN_b	wb (µm)	0.75	0.5	0.5
	lb (µm)	0.200	0.150	0.150
	nb	4	2	2
ib (µA)		38	45	46

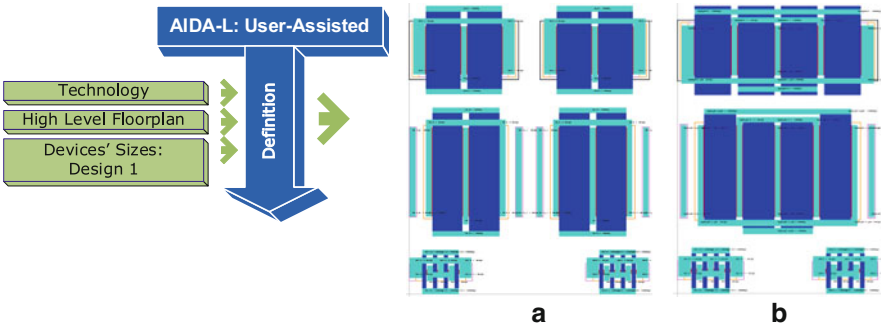


Fig. 4.14 Floorplan generated using the template file of Fig. 4.7 (*template 1*) and the devices' sizes of design 1: (**a**) only basic cells; (**b**) with cells combined

4.6.2 Retargeting Operation: Design 2

Then, the same *template 1* that contains the expert design knowledge for design 1 was used but with the set of devices' sizes from design 2. This is obviously a very simple example, but is important to notice that the retargeting operation for a different specification is made at push-button speed by reusing the previously designed template, and the result is presented in Fig. 4.15. Although all the devices' sizes were changed in this retargeting operation, is possible to change the dimensions of only a single device and promptly observe the result in the floorplan.

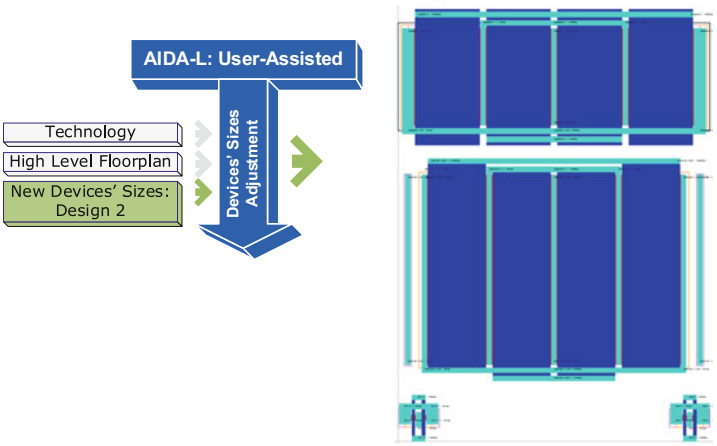


Fig. 4.15 Floorplan generated using the *template 1* and the devices' sizes of design 2: with cells combined. All floorplans are plotted at the same scale

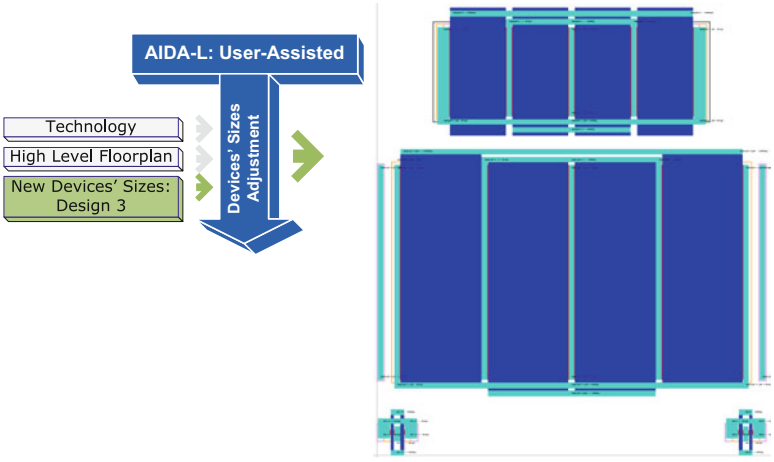


Fig. 4.16 Floorplan generated using the *template 1* and the devices' sizes of design 3, only cells combined considered. All floorplans are plotted at the same scale

4.6.3 Retargeting Operation: Design 3

Again, the same *template 1* was used but the devices' sizes of design 3 were provided instead. Design 3 has approximately three times the circuit area of design 1 and some cells have huge dimensions when comparing to the remaining, as presented in Fig. 4.16. Although the floorplan reflects the relative positioning between devices contained in the *template 1*, they may not yield the best floorplan possible.

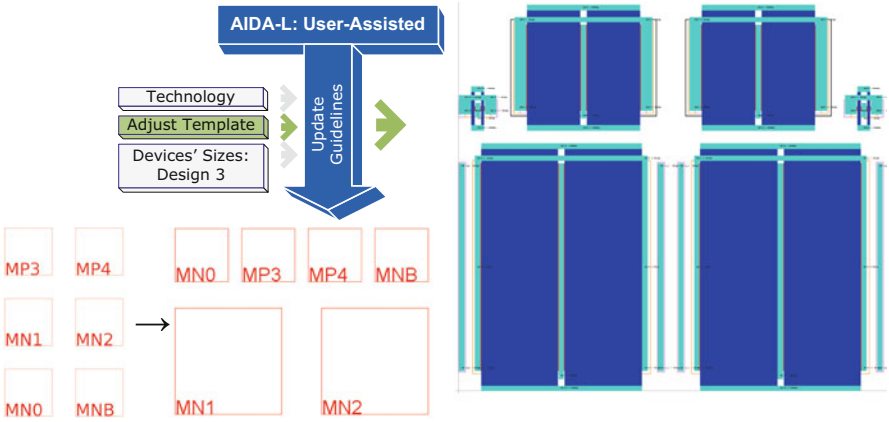


Fig. 4.17 Floorplan obtained when performing adjustments to the *template 1* and using the devices' sizes of design 3, only basic cells considered. All floorplans are plotted at the same scale

Small adjustments in the *template 1* can be made for a new relative positioning between cells as presented in Fig. 4.17, and a better packing is obtained. Since technological details are treated automatically by AIDA-L's template-based Placer, the designer is performing small and intuitive changes in this abstraction layer, the template file, refining his guidelines as suited, within seconds. While these retargeting operations are time consuming when performed in the traditional layout editing environments.

4.7 Conclusion

AIDA-L's template-based Placer, presented in this chapter, has the main purpose of creating an abstraction layer between designer's knowledge and technology details. The designer provides the high level floorplan and the tool instantiates and places the devices on the floorplan instantaneously, automatically taking biasing considerations.

The designer's guidelines, stored in the hierarchical XML template description, contain information about the relative positioning, rotation and topological constraints of each cell. As in the manual design, the high level floorplan inherits valuable information for the minimization of parasitic structures, smaller routing topology, special care for signal- or current-flow, etc., by placing devices in the desired locations. Therefore, knowing that the instantiated cells comply with the design rules and assuming that the circuit is properly biased, maintaining the distance between devices is the only operation required to verify the technology design rules.

The adopted B*-tree extraction and packing techniques were introduced. Since the extraction of the high level floorplan guidelines from the XML template file is inherently technology- and specification-independent, a simple amplifier was used in the end of the chapter to illustrate the possible obtained floorplans where the reusability of expert design knowledge and the efficiency on retargeting operations are implicit.

References

1. 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
2. A. Canelas, R. Martins, R. Póvoa, N. Lourenço, J. Guilherme, N. Horta, Enhancing an automatic analog IC design flow by using a technology-independent module generator, in *Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design*, ed. by M. Fakhfakh, E. Tlelo-Cuautle, M.H. Fino (IGI Global, Hershey, PA, 2014)
3. 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
4. T. Carusone, D. Johns, K. Martin, *Analog Integrated Circuit Design*, 2nd edn. (Wiley, New York, 2011)
5. Y. Yilmaz, G. Dundar, Analog layout generator for CMOS circuits. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **28**(1), 32–45 (2009)
6. M. Fakhfakh, E. Tlelo-Cuautle, M. Fino (eds.), *Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design* (IGI Global, Hershey, PA, 2014)
7. F. Maloberti, *Analog Design for CMOS VLSI Systems* (Kluwer, Boston, MA, 2001)
8. H.E. Graeb (ed.), *Analog Layout Synthesis: A Survey of Topological Approaches* (Springer, Berlin, 2010)
9. R. Martins, N. Lourenço, N. Horta, LAYGEN II—automatic layout generation of analog integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(11), 1641–1654 (2013). doi:10.1109/TCAD.2013.2269050
10. N. Lourenço, A. Canelas, R. Póvoa, R. Martins, N. Horta, Floorplan-aware analog IC sizing and optimization based on topological constraints. Integr. VLSI J. **48**, 183–197 (2015). Reprinted from Integration, the VLSI Journal. With permission from Elsevier

Chapter 5
Optimization-Based Placer

This chapter addresses AIDA-L's optimization-based Placer. Unlike, AIDA-L's template-based Placer, presented in Chap. 4 of this book, the optimization-based Placer dispenses most of the information contained in the template file. Instead, it applies a multi-objective (MO) algorithm to an absolute floorplan representation in order to determine the cells' locations. Cells can be organized in proximity groups which implement the desired set of symmetry and proximity requirements, bridging the difficulties found on the state-of-the-art works on fulfilling the proximity constraints. Moreover, and to reduce the problem complexity, the intrinsic hierarchy of analog integrated circuit (IC) layout is explored, where the Pareto fronts for each proximity group of the problem are combined in a bottom-up fashion.

Contrasting with the template-based Placer that outputs a single solution, in the absence of the high-level constraints it is extremely difficult to present a single, most meaningful and designer oriented, solution. The optimization-based Placer provides the designer with a Pareto front of placements representing the feasible tradeoffs between the optimization objectives. In addition, routing-related data, in the form of current-flow and current-density considerations, is also included in the optimization to improve circuit's routability, performance and post-layout reliability.

The first section of this chapter covers the general architecture of the optimization-based Placer. In section "Constrained Archive-Based Multi-Objective Simulated Annealing Algorithm", the developed MO algorithm is outlined, and, in section "XML Description for Optimization-Based Placement", the relevant information contained in the template file is described. In section "Hierarchical Placement Optimization in Absolute Coordinates", the problem of placement optimization with constraint handling in absolute coordinates is defined, as well as the MO hierarchical framework. In section "Current-Flow and Current-Density Considerations", the current-flow constraints and current-density considerations taken during placement optimization are detailed.

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_5

5.1 Optimization-Based Placer Architecture

The proposed architecture/design flow is shown in Fig. 5.1, which depicts the main tasks performed by the optimization-based Placer to generate the floorplan with only an extremely reduced set of guidelines. In the core of the optimization-based Placer is the multi-objective optimization (MOO) kernel, namely, the constrained archive-based multi-objective simulated annealing (CAMOSA) algorithm (Sect. 5.2). This Placer operates with a reduced setup (Sect. 5.3) over an absolute representation of the layout and, changes the cells' locations (Sect. 5.4) computing the amount of illegal overlap of each tentative solution.

Routing-related data, i.e., Current-Flow and Current-Density Considerations (Sect. 5.5), are taken to enforce the current-flow constraints directly in the abso-

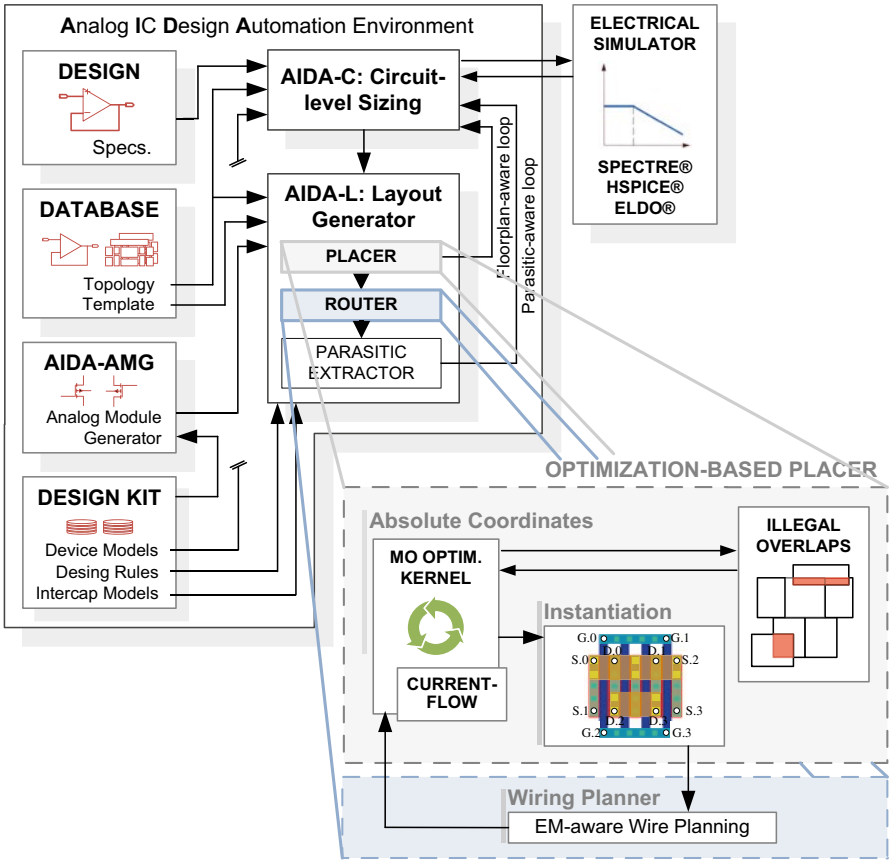


Fig. 5.1 Optimization-based Place and Route architecture: multi-objective optimization in absolute coordinates, instantiation and wiring planner blocks embedded in the AIDA-L's design flow

lute representation in order to reduce the solution space; while the current-density considerations are taken with the electromigration (EM)-aware minimization of the wiring topology (WT) for all the nets of the circuit during optimization. The detailed construction of the EM-aware WT is introduced later in Chap. 6 of this book. Furthermore, instantiation of the cells is required prior to the construction of each EM-aware WT, under the same basis previously introduced in Chap. 4 of this book.

5.2 Constrained Archive-Based Multi-Objective Simulated Annealing Algorithm

Before moving into the Placer details, first the developed constrained archive-based multi-objective simulated annealing algorithm is here described. The capabilities of population-based algorithms to maintain diversity, find multiple optima simultaneously or parallelism make them highly preferred choices for solving MOO problems. However, in the specific problem of analog placement, due to the large unfeasible region of the solution space it is especially important that the algorithm has the ability to accept unfavorable solutions and do not establish on local optima, i.e., around the first feasible solution founds. Since that a feasible solution does not necessarily mean that a good placement solution (compacted) was found.

Some of the desirable features of evolutionary population-based algorithms can be implemented with the use of an archive containing the best solutions found, while maintaining simulated annealing (SA) [1] the responsible for the evolution process. Early approaches to multi-objective simulated annealing (MOSA) combined several objectives into a single-objective function to construct the Pareto front [2–5], but, as proved in [6], parts of the fronts may be inaccessible using fixed weights. The archive-based multi-objective simulated-annealing (AMOSA) algorithm [7] is possibly the best true multi-objective adaption of SA found in the literature. It presents competitive results or outperforms other published multi-objective SA-based algorithms, and also, well known evolutionary algorithms such as NSGA-II [8] and PAES [9]. There, Pareto-dominance and amount of dominance concepts are used to determine the acceptance of a new solution, however, only unconstrained problems were considered.

The developed algorithm follows the guidelines of AMOSA and introduces several new concepts for handling constrained MOO problems. The CAMOSA algorithm is outlined on Algorithm 5.1 and the major changes in terms of dominance measures, annealing schedules and archive compaction are listed below.

Algorithm 5.1: Proposed constrained archive-based multi-objective simulated annealing algorithm [21]

define: T_{fmax}, T_{gmax}, R, k, *n_iterations*, *archivesize*

1. initialize Archive with random solutions
2. u = random solution from Archive
3. **for** ($t = 0$; $t < n\_iterations$, t++) **do**
4. v = move(u)
5. check constrained Pareto dominance(u, v)
6. **if** (u dominates v) **then**
7. set $u = v$ with a probability p:
8. Equation (5.12) with $\Delta fdom_{avg} = \dfrac{\left(\sum_{i=1}^{k} \Delta fdom_{i,v}\right) + \Delta fdom_{u,v}}{k+1}$ and $\Delta gdom_{avg} = \dfrac{-\left(\sum_{i=1}^{k} \Delta gdom_{v,i}\right) - \Delta gdom_{v,u}}{k+1}$
9. with $k \geq 0$ points in the Archive that dominate v
10. **if** (u and v are non-dominating to each other) **then**
11. **if** (v is dominated by $k \geq 1$ points in the Archive) **then**
12. set $u = v$ with a probability p:
13. Equation (5.12) with $\Delta fdom_{avg} = \dfrac{\left(\sum_{i=1}^{k} \Delta fdom_{i,v}\right)}{k+1}$ and $\Delta gdom_{avg} = -\dfrac{\left(\sum_{i=1}^{k} \Delta gdom_{v,i}\right)}{k+1}$
14. **if** (v is non-dominating w.r.t. Archive) **then**
15. set $u = v$
16. add v to the Archive
17. **if** (elements in the Archive > *archivesize*) **then**
18. assign crowding distance to the Archive and keep *archivesize* best elements
19. **if** (v dominates $k \geq 1$ points in the Archive) **then** //follow AMOSA procedure
20. set $u = v$
21. add v to the Archive and remove all the k dominated points from the Archive
22. **if**(v dominates u) **then**
23. **if** (v is dominated by $k \geq 1$ points in the Archive) **then**
24. $p = \dfrac{1}{1 + e^{[-\Delta dom_{\min}]}}$ with $\Delta dom_{\min}$ as the minimum of the difference of domination amounts $\Delta dom_k = \Delta fdom_{v,k} - \Delta gdom_{k,v}$ between v and the k points
25. set u = point k of the archive that corresponds to Δdom_{min} with a probability p, if not set $u = v$
26. **if** (v is non-dominating w.t.r. Archive) **then**
27. set $u = v$
28. add v to the Archive
29. **if** (elements in the Archive > *archivesize*) **then**
30. assign crowding distance to the Archive and keep *archivesize* best elements
31. **if** (v dominates k points in the Archive) **then** //follow AMOSA procedure
32. set $u = v$
33. add v to the Archive and remove all the k dominated points from the Archive

`Dominance Measures`

The acceptance probability of a new solution is determined by its feasibility status, and its domination status when compared to the current solution and the non-dominated solutions in the archive. In AMOSA, the traditional principle of objective-based dominance is used to compare two solutions *a* and *b*. If the relative performance of *a* is no worse for all objectives than the relative performance of *b*, and, better for one or more objectives, it is said that *a* dominates *b*, i.e., in a minimization problem *a* dominates *b* if:

$$f_i(a) \leq f_i(b) \quad \forall_i = 1,\ldots,n_{objectives} \tag{5.1}$$

$$f_i(a) < f_i(b) \quad for at least one i \tag{5.2}$$

In CAMOSA, in order to support constraints efficiently, the Pareto dominance is modified with the sum of constraint violation values *gsum* [8] and is here used as first criteria before objective-based domination of Eqs. (5.1) and (5.2). The constraint-based domination states that *a* dominates *b* if one of the following conditions (5.3) or (5.4) is true:

$$\left(g_{sum}(a) < 0\right) \& \left(g_{sum}(b) < 0\right) \& \left(g_{sum}(a) > g_{sum}(b)\right) \tag{5.3}$$

i.e., if both solution *a* and *b* are unfeasible, and, *a* is closer to feasibility than *b*;

$$\left(g_{sum}(a) = 0\right) \ \& \ \left(g_{sum}(b) < 0\right) \tag{5.4}$$

i.e., if solution *a* is feasible and solution *b* is unfeasible.

Using the same principle, *b* dominates *a* if one of the following conditions (5.5) or (5.6) is verified:

$$\left(g_{sum}(a) < 0\right) \& \left(g_{sum}(b) < 0\right) \& \left(g_{sum}(a) < g_{sum}(b)\right) \tag{5.5}$$

i.e., if both solution *a* and *b* are unfeasible, and, *b* is closer to feasibility than *a*;

$$\left(g_{sum}(a) < 0\right) \& \left(g_{sum}(b) = 0\right) \tag{5.6}$$

i.e., if solution *a* is unfeasible and solution *b* is feasible.

If none of the above conditions are verified, the solutions are considered non-dominating with respect to constraints, and then the second criteria, objective-based domination, is considered. In the original algorithm, the concept of amount of dominance is used to compute the probability of acceptance of a new point. In the proposed approach, this concept is extended into two different dominance measures.

Given two solutions a and b, the amount of objective dominance $\Delta fdom_{a,b}$ is defined by:

$$\Delta fdom_{a,b} = \prod_{i=1, f_1(a) \neq f_1(b)}^{n_{objectives}} \frac{\left| f_1(a) - f_1(b) \right|}{R_i} \tag{5.7}$$

where $n_{objective}$ is the number of objectives being optimized and R_i is the range of the objective i found so far. R_i can be computed using the objective values of the solutions contained in the archive, and also, the new and current solution. The amount of objective-based dominance is defined in terms of hypervolume of function space enclosed by the points being compared.

In the second concept, given two solutions a and b, the amount of constraint dominance $\Delta gdom_{a,b}$ is defined by:

$$\Delta gdom_{a,b} = g_{sum}(a) - g_{sum}(b) \tag{5.8}$$

where g_{sum}, as stated previously, is the sum all constraint violation values. For placement optimization, if only illegal overlaps are considered as constraint, it is equivalent to:

$$\Delta gdom_{a,b} = g_0(a) - g_0(b) \tag{5.9}$$

This way, in CAMOSA, the acceptance probability depends on both the amount of objective and constraint domination.

5.2.1 Double Annealing Schedule

Another issue to account in weightless SA-based constrained optimization is that, objectives and constraints can have completely different magnitudes, e.g., while minimizing placement area $f_0(x)$ constrained to illegal overlaps $g_0(x)$ equal to zero, it is usual to have huge placement area values when comparing to feasible or close to feasibility solutions, $f_0(x) \gg g_0(x)$. For this reason two different annealing temperatures for objectives and constraints are considered, defined as:

$$T_f(t) = T_{f\max}.e^{-R.\left(\frac{t}{k}\right)} \tag{5.10}$$

$$T_g(t) = T_{g\max}.e^{-R.\left(\frac{t}{k}\right)} \tag{5.11}$$

where T_{fmax} is the maximum (initial) temperature for annealing the objectives factor and T_{gmax} the maximum (initial) temperature for annealing the constraint factor. Even though the annealing schedules are the same, this allows using non-normalized

values for the performance and constraint measures. With this double annealing schedule the probability of acceptance of new solution is defined as:

$$p = \frac{1}{1 + e^{\Delta fdom_{avg} * T_f(t) + \Delta gdom_{avg} * T_g(t)}} \tag{5.12}$$

where $\Delta fdom_{avg}$ and $\Delta gdom_{avg}$ are the average amounts of objective and constraint domination, respectively. These average amount values are computed with different parameters according to the dominance status of the new solution with respect to the current solution and solutions in the archive, lines 8 and 13 of Algorithm 5.1.

5.2.2 Archive Compaction

An archive of non-dominated solutions found during the search is maintained, which will form the final Pareto front of placements. While the archive is randomly initialized, once a non-dominated solution is archived the remaining dominated solutions are removed. In AMOSA, clustering by single linkage is applied to reduce the number of the solutions in the archive to a maximum fixed value. In the proposed approach, crowding distance concept is applied to the archive of non-dominated solutions and then the *archivesize* best placements are kept. This operator is similar to the one used in NSGA-II [8], and attempts to mitigate the loss of diversity as the number of solutions in the Pareto front is reduced.

5.3 XML Description for Optimization-Based Placement

The main purpose of the optimization-based Placer is to reduce the setup, nevertheless some setup is still considered. The template information that is used is the type of the cells, and the symmetry, matching and combine requirements. Since the template file is already semi-automatically generated from the parameterized netlist, with the type of cell and matching constraints inferred, only the symmetry and combines properties must actually be provided by the designer.

The complete template required by the optimization-based Placer for the simple differential amplifier introduced in Chap. 4 of this book is presented in Fig. 5.2. When compared to the XML description for the template-based Placer, the relative positioning of each cell and rotate operators are obsolete.

In this work, the symmetry setup is made manually. However, there are works in the literature that focus on the extraction of placement rules, e.g., Eick et al. [10] presents a method to automatically generate hierarchical placement rules, where several types of proximity, matching and symmetry constraints are determined. With the implementation of a similar method, and since the parameterized netlist contains all the required information, the template file can be easily discarded and further alleviate the designer's contribution during setup.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Template SYSTEM "template4.dtd">
<Template name="TMP_NAME">
    <CellList>
        <Cell name="MP3" symGroupId="1" symCellId="1">
            <MOSFET type="P" width="_W3" length="_L3" nf="_NF3"/>
        </Cell>
        <Cell name="MP4" symGroupId="1" symCellId="1">
            <MOSFET type="P" width="_W3" length="_L3" nf="_NF3"/>
        </Cell>
        <Cell name="MN1" symGroupId="1" symCellId="2">
            <MOSFET type="N"  width="_W1" length="_L1" nf="_NF1"/>
        </Cell>
        <Cell name="MN2" symGroupId="1" symCellId="2">
            <MOSFET type="N"  width="_W1" length="_L1" nf="_NF1"/>
        </Cell>
        <Cell name="MN0" symGroupId="1" symCellId="3">
            <MOSFET type="N"  width="_WB" length="_LB" nf="_NFB"/>
        </Cell>
        <Cell name="MNB" symGroupId="1" symCellId="3">
            <MOSFET type="N"  width="_WB" length="_LB" nf="_NFB"/>
        </Cell>
    </CellList>
</Template>
```

Fig. 5.2 XML template file introduced in Chap. 4 of this book, but for the optimization-based Placer. Note that the only guidelines provided by the designer are *symGroupId* and *symCellId* fields

5.4 Hierarchical Placement Optimization in Absolute Coordinates

In this section, the proposed formulation for the problem of hierarchical placement optimization in absolute coordinates is provided. The notation used henceforward is sketched on Table 5.1.

5.4.1 Analog Constraints and Proximity Groups

Layout parasitics, process variations and thermal gradients have a huge impact on the on-die circuits' performance, and to handle them, the whole analog design is subject to an exhaustive set of stringent constraints. Particularly, symmetry constraints, which are mandatory in the design of differential circuits to guarantee an identical behavior of the two symmetric parts of the circuit, must be imposed not only to the devices' placement, but also to the interconnects between them. In

Table 5.1 Notation used

Notation	Definition
c	Cell
P	Proximity group
(c^1, c^2)	Symmetry pair
c^s	Self-symmetric cell
c^a	Autonomous cell
(x_i, y_i)	Bottom left coordinate of the cell c_i
w_i	Width of the cell c_i
h_i	Height of the cell c_i
r_i	Rotation angle of the cell c_i
$\ddot{x}_i$	x Coordinate of the vertical symmetry axis of the proximity group P_i
k_i	Representation of a proximity group P_i

placement, as depicted in Fig. 5.3, symmetry constraints are formulated in two ways, for each symmetry axe:

- Symmetry pair (c^1, c^2)—two matched cells placed symmetrically in relation to the vertical symmetry axis, according to Eqs. (5.13) and (5.14):

$$\frac{1}{2}\left(x_i^1 + \frac{w_i}{2} + x_i^2 + \frac{w_i}{2}\right) = \ddot{x}_i \tag{5.13}$$

$$y_i^1 + \frac{h_i}{2} = y_i^2 + \frac{h_i}{2} \tag{5.14}$$

  e.g., symmetry pairs $(c_1{}^1, c_1{}^2)$ and $(c_2{}^1, c_2{}^2)$ in Fig. 5.3;

- Self-symmetric cell c^s—a cell that is placed centered in the vertical symmetry axis, e.g., cells $c_3{}^s$ and $c_4{}^s$ in Fig. 5.3.

The definitions/equations for a horizontal symmetry axis can be defined by analogy.

It is known that with the increase of distance between the two cells of a symmetry pair, the difference between their electrical properties also increases due to the inherent mismatch from the fabrication process. In order to place the devices within a symmetry pair in close proximity minimizing the sensitivities to process variations, the concept of symmetry group containing only symmetric cells was introduced in [11]. In this approach, an improved version is considered:

- Proximity Group S—contains one symmetry axis and is the compact placement of any number of symmetry pairs, self-symmetric modules and unlike previous symmetry-group-based works, autonomous cells, i.e., cells without symmetry requirements, e.g., cell $c_5{}^a$ in Fig. 5.3, which represents k_i, one possible compact placement for the proximity group P. Therefore, within the same proximity

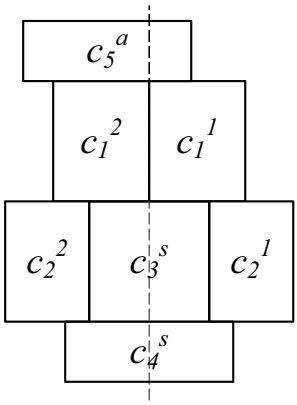


Fig. 5.3 Illustration of a possible representation, k_i, for the proximity group P, containing different cell types: symmetry pairs c_1 and c_2, self-symmetric cells c_3 and c_4, and autonomous cell c_5

group, proximity constraints can be forced by assigning cells to be placed closely, with or without symmetry requirements.

For instance, in a CMOS cell it is usually desired to have all PMOS and NMOS devices placed closely in two distinct groups. As overviewed in Chap. 2 of this book, previous works penalize, in the cost function, solutions with symmetric modules far from each other. Furthermore, as most of the reviewed approaches do not support hierarchical clustering, proximity constraints are not supported efficiently as cells without symmetry requirements could be placed far away from the rest of the group during as a result from the optimization process. Hierarchical B*-tree (HB*-tree) [11–13] and enhanced hierarchical Slicing-Tree (HS-tree) [14] are the only topological representations that can guarantee the closest proximity of symmetric modules within a symmetry group, however, the automatically symmetric-feasible B*-trees/Slicing-trees that optimize the placement of the symmetry groups do not support cells without symmetry requirements.

Without a group-based approach that supports any cell type, the remaining proximity constraints, e.g., placement of each PMOS/NMOS group, can only be dealt with additional hierarchy levels. The complexity of the design hierarchy definition can escalate quickly for more complex circuits, and while the optimization algorithm is responsible to optimize the floorplan, the design hierarchy is specified by the designer. In the proposed approach this fake hierarchy is removed as proximity groups with any cells, e.g., for well or guard ring sharing, are obtained directly from the representation.

Consider the single ended two-stage operational amplifier of Fig. 5.4 and the imposed symmetry (which implies matching) and proximity requirements for the current design. In Fig. 5.5 is represented the design hierarchy required to implement the set of constraints using a symmetry-group-based approach, while in Fig. 5.6 it is sketched the design hierarchy for the proposed proximity-group-based approach.

In the provided example the advantage of considering autonomous cells within symmetry groups is relevant for n-well and guard ring sharing. In the architectures with bad handling of proximity constraints, cells with no special symmetry requirements are often improperly placed as self-symmetric cells, however, as self-symmetric

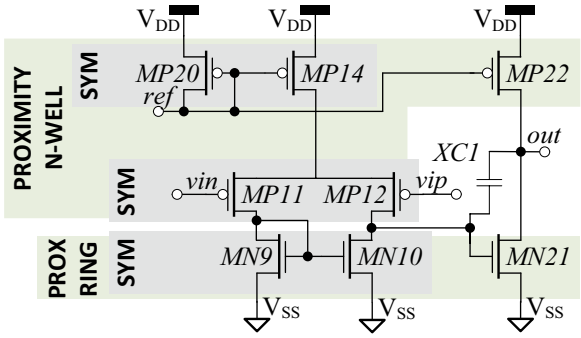


Fig. 5.4 Single ended two-stage amplifier schematic with highlighted symmetry and proximity requirements [21]

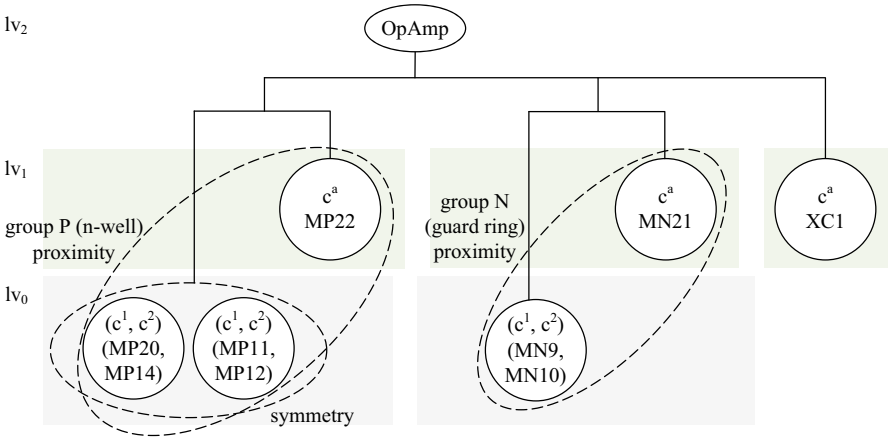


Fig. 5.5 Design hierarchy required when using a symmetry-group-based approach. Proximity constraints between cells with symmetry requirements and cells without symmetry requirements are satisfied using extra hierarchy levels [21]

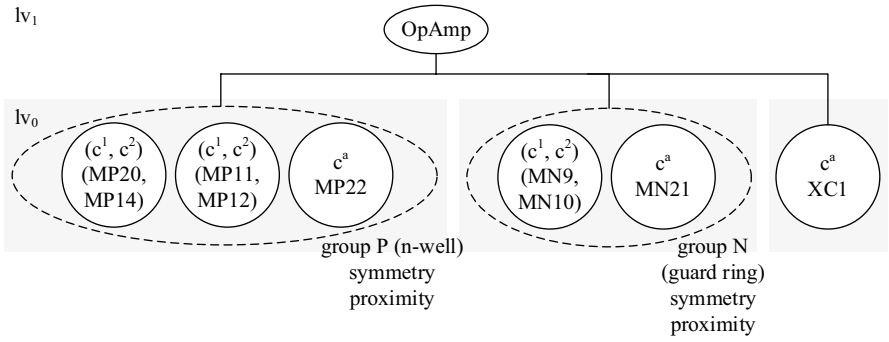


Fig. 5.6 Design hierarchy using the proposed proximity group-based approach [21]

cells are highly constrained cells, that design option always conditions the quality of the resulting floorplan.

5.4.2 Absolute Coordinates' Problem Definition

Due to the nature of the problem, the optimizer moves the coordinates of each cell directly. In order to reduce the number of optimization variables and force the desired symmetries explicitly, in a symmetry pair $\left(c_i^1, c_i^2\right) = \left(\left(x_i^1, y_i^1\right), \left(x_i^2, y_i^2\right)\right)$ only one pair of bottom left coordinates (x_i, y_i) is moved, the other is found deterministically from the first, according to:

$$\begin{aligned} x_i^1 &= \ddot{\ddot{x}} + x_i \\ x_i^2 &= \ddot{\ddot{x}} - x_i - w_i \\ y_i^1 &= y_i^2 = y_i \end{aligned} \tag{5.15}$$

And for a self-symmetric cell:

$$\begin{aligned} x_i^s &= \ddot{\ddot{x}} - \left(w_i / 2\right) \\ y_i^s &= y_i \end{aligned} \tag{5.16}$$

Again, the same analogy can be used for a horizontal symmetry axis without loss of generality.

Proximity groups, after being optimized at the bottom level of the hierarchy, are moved as self-symmetric, as part of a symmetry pair or as an autonomous cell in the top levels of the hierarchy. The movable reference point is the bottom left coordinate of the rectangular bounding box that contains all the cells, as depicted in Fig. 5.7a. However, since the proximity group is defined by the cells contained in it, the contour is a rectilinear structure which allows other cells to be placed in a non-rectangular shape, as depicted in Fig. 5.7b. These groups, besides reducing the number of variables (the search space) at each hierarchy level, also ease handling advanced analog constraints such as alignment between any number of cells or symmetry between groups or proximity between groups, etc.

To summarize, the number design variables required for each cell type is sketched on Table 5.2. It is important to notice that in some analog applications the rotation angle of device modules may not be allowed to change, which will further reduce the number of variables.

As in the traditional absolute coordinate placers, the amount of illegal overlap is computed by the interception of the bounding box of each cell, as illustrated in Fig. 5.7c, d. The essential constraint considered is the area of illegal overlap, whose value must be driven to zero in order to obtain a feasible solution:

$$g_0\left(x\right) = -overlap \tag{5.17}$$

For example, in Fig. 5.7c, d, although both solutions are unfeasible solutions as $g_0\left(c\right) < 0$ and $g_0\left(d\right) < 0$, the solution (d) is worse than the solution (c) because the illegal overlap area is higher, i.e., $g_0\left(d\right) < g_0\left(c\right)$.

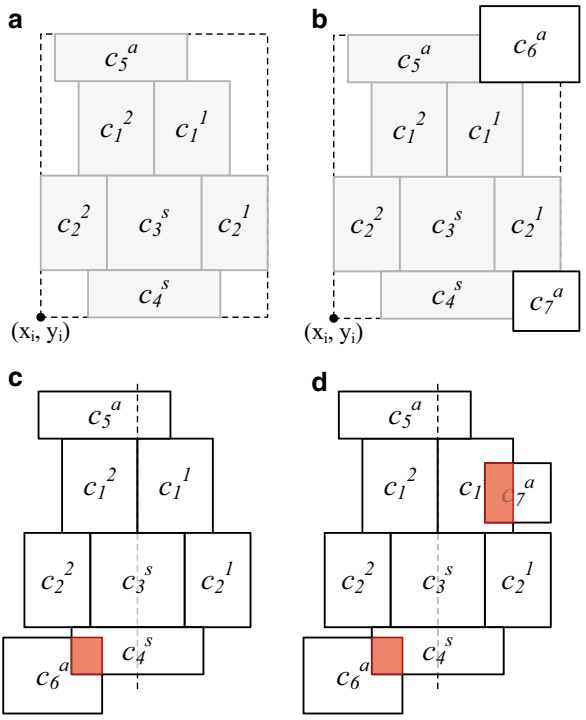


Fig. 5.7 Movable proximity group of Fig. 5.3: (**a**) reference point for moving the quadrangular bounding box of the proximity group; (**b**) cells c_6 and c_7 do not belong to the proximity group but can be placed closely in a non-rectangular shape. (**c**, **d**) Illegal overlaps (area of overlap marked at *red*), $g_0(c) > g_0(d)$

Table 5.2 Parameters optimized for each cell type

Cell type	#Optimization variables
Symmetry pair (c^1, c^2)	x_i, y_i, r_i
Self-symmetric cell c^s	y_i, r_i
Autonomous cell c^a	x_i, y_i, r_i
Proximity group P (as autonomous)	x_i, y_i, r_i, k_i

5.4.3 Multi-Objective Hierarchical Framework

In the automatic placement approaches the number of design variables escalates quickly with the number of devices, and, leads to a huge dimension of the design variables' vector x^d, given by Eq. (5.6), for optimization.

$$x^d = \sum_{sp}\left(x_i,,y_i,,r_i\right) + \sum_{s}\left(y_i,r_i\right) + \sum_{a}\left(x_i,,y_i,,r_i\right) \tag{5.18}$$

where *sp*, *s* and *a* are the number of symmetry pairs, self-symmetric and autonomous cells of the circuit, respectively. Analog floorplan has a hierarchical nature, and, this hierarchy can be used to reduce the problem complexity through different executions of the MOO kernel. In the proposed hierarchical framework, partitions and any cell type are combined in a bottom-up manner in order to reduce the

number of design variables at each optimization stage but maximizing the number of solutions provided to the hierarchy levels above.

Since multiple floorplan solutions in the form of Pareto fronts are passed bottom-up through the design hierarchy, the floorplan quality at higher levels is increased. Consider the design hierarchy introduced in Fig. 5.6, the optimization outputs of each partition, i.e., resultant Pareto fronts of placements, of the lower levels are obtained first and then combined at the highest level, as illustrated in Fig. 5.8.

This way, in the current case, at the highest level (*OpAmp*) only small number of variables $x^{d}{}_{OpAmp}$ is optimized:

$$x^{d}_{OpAmp} = \left(x,,,y,,,r,,,k\right)_{groupP} + \left(x,,,y,,,r,,,k\right)_{groupN} \tag{5.19}$$

where the k_i parameters define which points of the Pareto fronts (placement solution) of the partitions of the hierarchy level below are currently selected. Notice that each partition can contain any number of sub-partitions using the same set of operations recursively.

5.5 Current-Flow and Current-Density Considerations

The attainable routing quality, as well as most parasitic effects and consequent post-layout circuit's performance degradation, are set once a placement solution is fixed. The desired reliability and performance may not be achieved post-layout only by minimizing the circuit's area and/or other objectives, such as total wire-length, while satisfying a set of matching, symmetry and proximity constraints. As more information pertinent to the routing must also be accounted during placement automation current-flow and current-density considerations were recently brought in [15], and are also considered in the proposed optimization-based Placer.

5.5.1 Current-Flow Constraints

Current-flow constraints are fulfilled with a monotonic routing of the critical electric-current/signal-paths, which was proved in [16] to reduce both the interconnect wirelength and routing-induced parasitics, improving the post-layout circuit's performance. The amplifier topology using voltage combiners proposed in [17] for the United Microelectronics Corporation (UMC) 130 nm design process is presented in Fig. 5.9a with the current-paths, corresponding to the current-flow constraints, i.e., paths from power to ground lines listed in Table 5.3, superimposed. Figure 5.9b, c show two different placements for the circuit with the current-paths also overlaid.

As suggested by Wu et al. [14], the current-flow constraints can be satisfied by forcing a monotonic routing on the desired current/signal-paths. Given a location

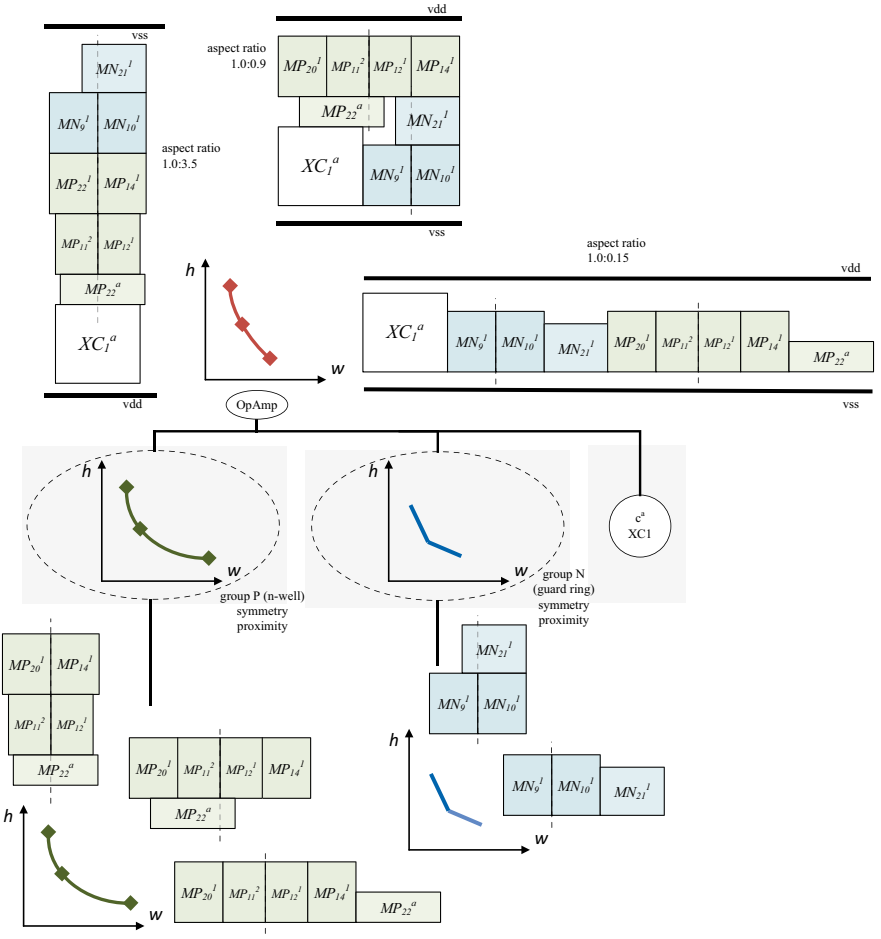


Fig. 5.8 For the design hierarchy of the single ended two-stage amplifier of Fig. 5.6, Pareto fronts with the tradeoff placement's height (h) versus width (w) of each proximity group are combined bottom-up through the hierarchy. For illustration purposes, the tradeoff h versus w is explored, but any number of objectives could be considered [21]

for the power/ground lines and in order to obtain a monotonic routing of the cells contained in a current-flow constraint, cells' should be placed according to the monotonic directions depicted in Fig. 5.10. As illustrated on Fig. 5.9b, c, while both placements have the same placement area, the placement (c) contains non-monotonic current-paths that will lead to a complex and longer routing topology and consequently larger routing-induced parasitics.

In this work, the current-flow constraints and power/ground lines' location are used to construct a set of topological relations between modules. For the set of current-flow constraints of Table 5.3 and the power/ground lines placement of Fig. 5.10a, the constraint-graph generated for the problem is depicted in Fig. 5.11.

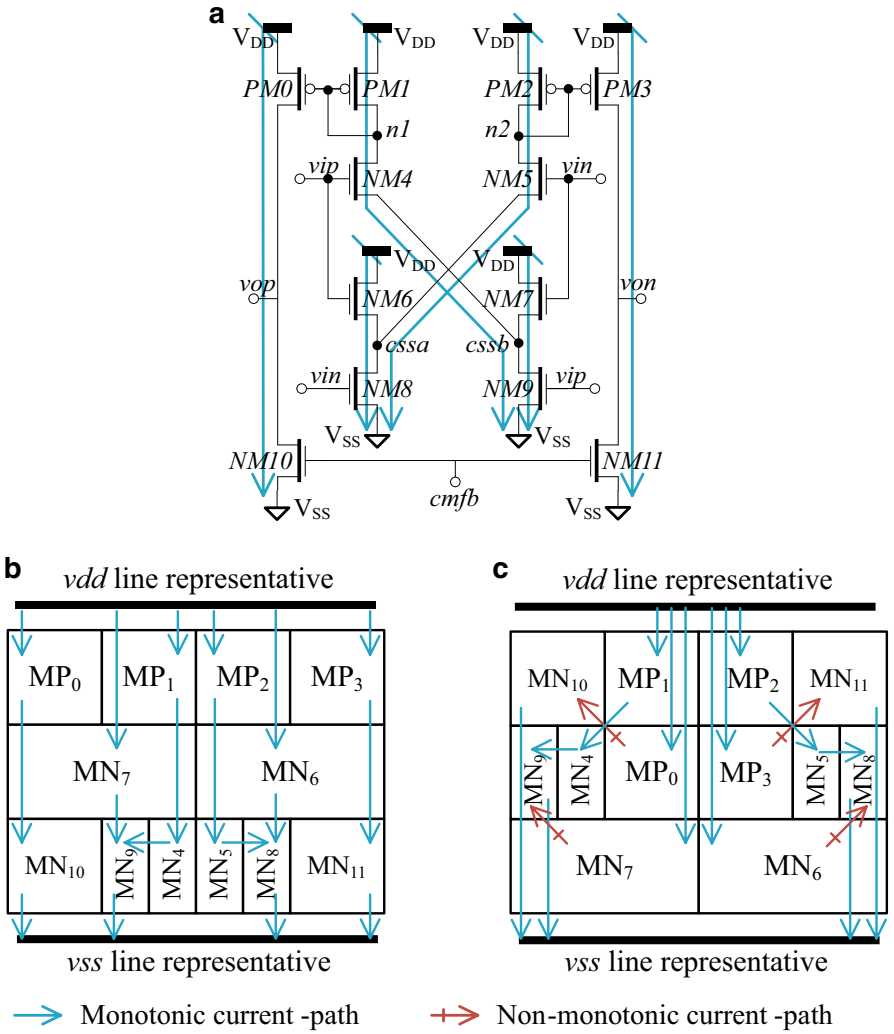


Fig. 5.9 Single stage amplifier with gain enhancement [17]: (**a**) schematic with current-paths highlighted; (**b**) Placement solution with monotonic current-paths; and (**c**) Placement solution with non-monotonic current-paths, i.e., that violate the current-flow constraints [20]

Table 5.3 Current-flow constraints for layout placement of the single stage amplifier

Current-flow constraints
$V_{dd} \rightarrow \mathrm{PM}_0 \rightarrow \mathrm{NM}_{10} \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{PM}_1 \rightarrow \mathrm{NM}_4 \rightarrow \mathrm{NM}_9 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{NM}_6 \rightarrow \mathrm{NM}_8 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{PM}_2 \rightarrow \mathrm{NM}_5 \rightarrow \mathrm{NM}_8 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{NM}_7 \rightarrow \mathrm{NM}_9 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{PM}_3 \rightarrow \mathrm{NM}_{11} \rightarrow V_{ss}$

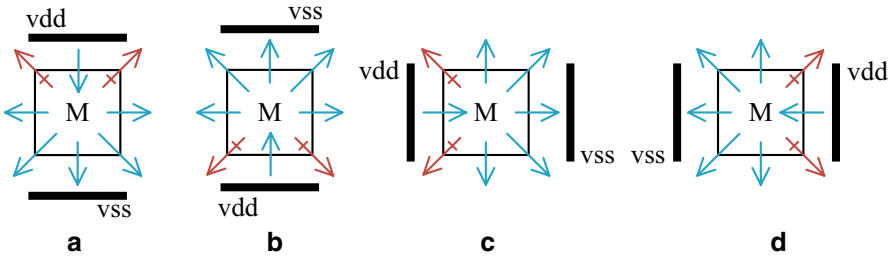


Fig. 5.10 Admissible monotonic routing for the different power and ground lines dispositions: (**a**) vdd line on top and vss line on bottom; (**b**) vdd line on bottom and vss line on top; (**c**) vdd line at left and vss line at right; and (**d**) vdd line at right and vss line at left [20]

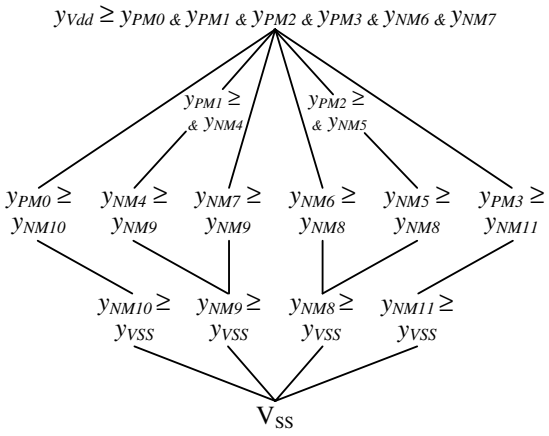


Fig. 5.11 Constraint-graph that relates the y's-coordinates of each cell, constructed for the set of current-flow constraints of Table 5.3 and the power/ground lines placement of Fig. 5.10a. This constraint-graph could represent the placement of Fig. 5.9b [20]

In this case, the restrictions are made referring the *y*-coordinate of each cell, i.e., defining a minimum allowed *y*-value and leave *x*-coordinate unconstrained. However, the same analogy can be used for the dispositions of Fig. 5.10c, d, where the restrictions would be made relating the *x*'s-coordinates.

The generated constraint-graph is forced over the absolute representation introduced in the previous section. The solution space is reduced as the design space for the cells' contained in the constraint-graph is restricted to only feasible regions of the y-coordinate (for the current case).

5.5.2 *Current-Density Considerations*

Current-density constraints are commonly taken by widening current-intensive nets to avoid the failure of the circuit due to EM, and it must be taken into account in the design of both power networks and signal wires, in order to ensure that the impact of EM effects on the circuits' reliability negligible. In addition, an optimal EM-aware WT for all nets is essential to improve the routing quality by reducing the wirelength of current-intensive (wider) interconnects.

However, the quality of the attainable WT is strongly related with the quality of the floorplan, hence, it is of vital importance to take the WT into consideration while placing the devices. In the proposed placer, given a floorplan, the correspondent netlist and a set of electric-currents for each device terminal, the electric-current-correct tree that provides the optimal terminal-to-terminal connectivity and flows, and that minimizes the wiring area is computed for each of the circuit's nets using the construction of the EM-aware WT method that will be properly detailed in Chap. 6 of this book.

The *EM-aware WT Problem Formulation* states that given a set of k terminals of a net, where p terminals are considered as current-sources (positive DC electric-current value) and q are current-sinks (negative DC electric-current value), the EM-aware WT is constructed by minimizing the total wire area of Eq. (5.20):

$$\sum_{i=1}^{p}\sum_{j=1}^{q} l_{ij} \times w_{ij}\left(I_{ij}\right) \tag{5.20}$$

where l_{ij} is the length and w_{ij} the effective wire's width that satisfies the EM for the root mean square current I_{ij} assigned to the wire ij in the WT. Summarizing, a higher current-density I_{eq} will require an increase in the interconnect's width in order to maintain the same median time to failure [18]. Using the previous concepts, the EM-aware WT for some nets (for illustration purposes) using the floorplan of Fig. 5.9b are presented in Fig. 5.12.

5.5.3 Application in the Hierarchical Framework

The optimization of the amplifier of Fig. 5.9a is used to illustrate the complete process, it was divided into two partitions, one containing the PMOS and the other, the NMOS devices. Symmetries, which imply matching, are imposed as in the original design between pairs of devices: MP_0/MP_3, MP_1/MP_2, MN_6/MN_7, MN_4/MN_5, MN_8/MN_9 and MN_{10}/MN_{11}. The optimization outputs of each partition, i.e., resultant set of Pareto non-dominated placements (for any number of objectives), of the lower levels are obtained first and then combined at the highest level, as illustrated in Fig. 5.13.

The current-flow constraints are partially satisfied at each partition with the available cells, and fully verified at the top partition (when all the cells are available). At the highest level the sum of the optimal EM-aware WT for all the nets of the circuit is minimized according to Eq. (5.21), where n is the number of nets of the circuit.

$$\min \sum_{l=1}^{n}\left(\sum_{i=1}^{p}\sum_{j=1}^{q} l_{ij} \times w_{ij}\left(I_{ij}\right)\right) \tag{5.21}$$

The WTs are only constructed at the top level of the hierarchy because the polynomial-complexity method used requires that the Kirchhoff's current laws are met in every terminal (Chap. 6 of this book). While the workaround, that is used for

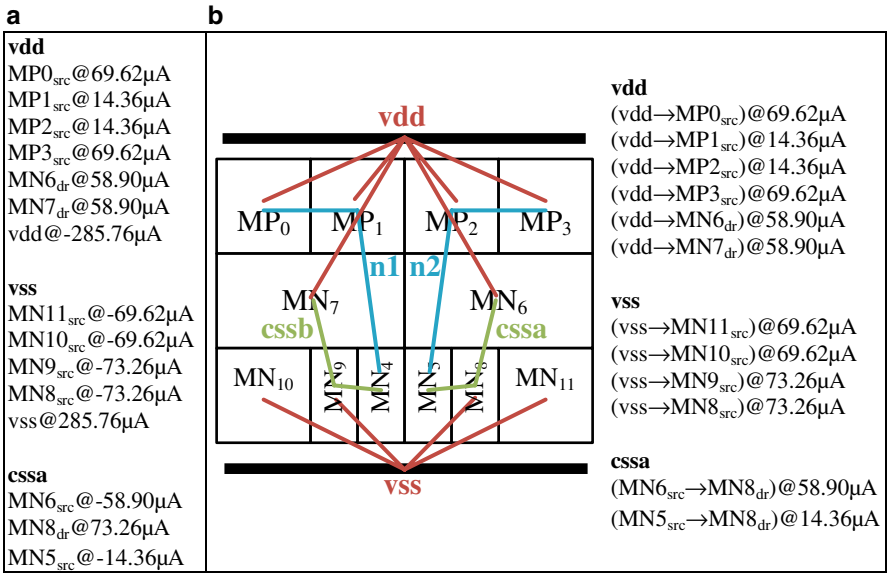


Fig. 5.12 (**a**) Netlist and generic electric-currents associated to each terminal (for only three nets); (**b**) EM-aware wiring topology. The wires' widths are function of the electric-current imposed on them [20]

the circuit terminals, could be used at partition level, it degrades the performance of the method, as it assumes an "ideal" auxiliary source (sink) terminal that will supply (absorb) any lack (excess) of current. "Ideal" is used here in the sense that the connection from other terminals to the new one is made with wires of length zero.

5.6 Conclusion

Without the high level floorplan guidelines for placement, the automation level is increased and the designer's effort reduced. However, to maximize the number of solutions provided to the designer, in the proposed Placer, MOO concept was applied to analog floorplan automation, with an improved exploration of the feasible regions of the solutions space with the proposed CAMOSA algorithm. No empirical and error-prone weights are required in the cost function to minimize several requirements and overlap penalty. In addition, the intrinsic analog hierarchy is used to define partitions that allow an efficient reduction of the problem complexity and, also, deals with analog proximity constraints, which are barely supported in previous approaches. The proposed optimization-based Placer provides a full set of solutions to the designer supporting all the commonly used floorplan constraints, and, without over-constraining the solution space with a topological representation that require packing, structural scan or post-processing methods fulfill the analog constraints.

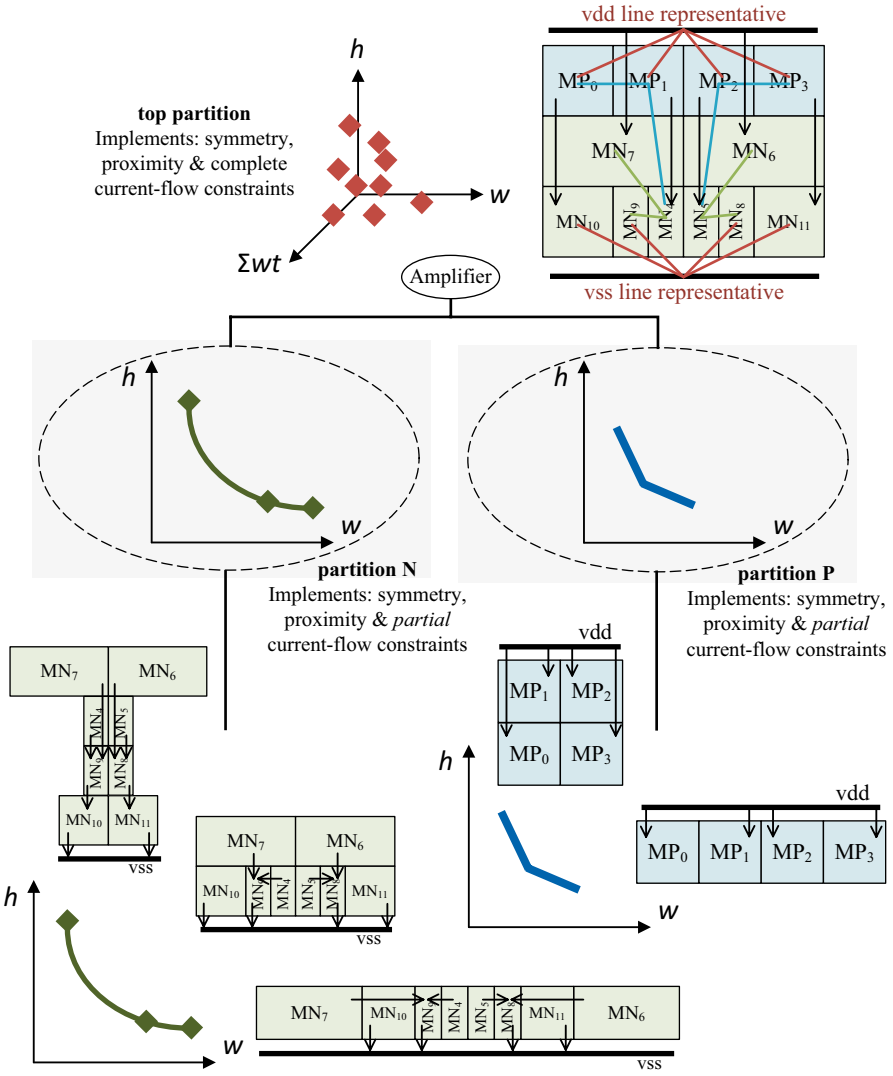


Fig. 5.13 Design hierarchy: Pareto fronts with the tradeoff placement's height (h) versus width (w) of each partition, that partially fulfill the current-flow constraints, are combined bottom-up through the hierarchy. For illustration purposes, the tradeoff h versus w is explored, but any number of objectives could be considered. At top partition, the minimization of the sum of the optimal EM-aware WT for all the nets is introduced as objective [20]

As overviewed in Chap. 2 of this book, modern analog placement algorithms aim to minimize the cell area while satisfying a set of symmetry, proximity, and other placement constraints. While the generated floorplans reflect the packing in terms of circuit area or aspect ratio for the provided modules, any kind of indicator of the circuit performance or routability not always is included during this process. When

routing-related data is incorporated into automatic placement, it is usually in the form of half-perimeter wirelength (HPWL) minimization, however, HPWL is not a reliable metric. As proved in [16], placements with longer HPWL may even lead to better circuit performances. Furthermore, it is known that to achieve better circuit performance and reduce the layout-induced parasitic impacts, it is desirable to simultaneously consider current-flow and current-density constraints at both placement and routing stages of analog layout synthesis [15].

Traditionally, current-density constraints are dealt separately, only at the routing stage, a detailed overview of the state-of-the-art of analog current-driven routing approaches was presented in Chap. 2 of this book. Going back to the few existent current-driven placement approaches, still, in [14, 16, 19] no current-density considerations are taken. While in [15], the routing phases are time consuming operations in-loop which strongly degrade the execution time, and also, with the simultaneously packing of the HB*-tree and routing (device by device) the global optimality of the attainable WT is lost. In the proposed methodology the current-flow constraints are used to reduce the solution space of the absolute representation. Routing and current-density considerations are taken, and the optimal (minimum) EM-aware WT for all the nets of the circuit, given the placement under evaluation, is computed and considered in the evaluation of each tentative placement. This will push for the shortening of current-intensive interconnects. These contributions are sketched in Table 5.4.

References

1. 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)

Table 5.4 State-of-the-art on current-flow and current-density-aware placement

Work	Structure	Algorithm	Included			Time (s)[a]
			Current-flow	Current-density	Routing	
Long et al. [19]	Linear programming (LP)		Straight	✗	✗	n/a
Ou et al. [15]	Enhanced HB*-tree	SA and dynamic programming	Monotonic	✓	*Minimize* global Router	~256[b]
Wu et al. [14, 16]	HS-tree	DeFer and LP	Monotonic	✗	*Minimize* HPWL	<1[c]
AIDA-L's optimization-based Placer [20, 21]	Absolute	Enhanced AMOSA	Monotonic	*Minimize* optimal EM-aware WT		~122[d]

[a]Reference computational times, using as benchmark a typical analog cell with less than 15 modules
[b]On an Intel® Xeon E5-2620 2.0 GHz with 72 GB of RAM
[c]On a 2.9 GHz Linux machine with 32 GB of RAM
[d]On an Intel® Core™ i7 2.3 GHz with 32 GB of RAM

2. D.K. Nam, C.H. Park, Multiobjective simulated annealing: A comparative study to evolutionary algorithms. Int. J. Fuzzy Syst. **2**(2), 87–97 (2000)
3. M. Hapke, A. Jaszkiewicz, R. Slowinski, Pareto simulated annealing for fuzzy multi-objective combinatorial optimization. J. Heuristics **6**(3), 329–345 (2000)
4. A. Suppapitnarm, K.A. Seffen, G.T. Parks, P.J. Clarkson, A simulated annealing algorithm for multiobjective optimization. Eng. Optim. **33**(1), 59–85 (1999)
5. E.L. Ulungu, J. Teghaem, P. Fortemps, D. Tuyttens, MOSA method: A tool for solving multiobjective combinatorial decision problems. J. Multi-Criteria Decis. Anal. **8**(4), 221–236 (1999)
6. I. Das, J. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Struct. Optim. **14**(1), 63–69 (1997)
7. S. Bandyopadhyay, S. Saha, U. Maulik, K. Deb, A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Trans. Evol. Comput. **12**(3), 269–283 (2008)
8. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
9. J. Knowles, D. Corne, The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization, in *Proceedings of the 1999 Congress on Evolutionary Computation*, July 1999, pp. 98–105
10. M. Eick, M. Strasser, K. Lu, U. Schlichtmann, H. Graeb, Comprehensive generation of hierarchical placement rules for analog integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **30**(2), 180–193 (2011)
11. 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
12. 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
13. 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)
14. 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. Circ. Syst. **33**(6), 879–892 (2014)
15. 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
16. P.-H. Wu, M. Lin, Y.-R. Chen, B.-S. Chou, T.-C. Chen, T.-Y. Ho, B.-D. Liu, Performance-driven analog placement considering monotonic current paths, in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, Nov 2012, pp. 613–619
17. R. Póvoa, N. Lourenço, N. Horta, R. Santos-Tavares, J. Goes, Single-stage amplifiers with gain enhancement and improved energy-efficiency employing voltage-combiners, in *21st IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Oct 2013, pp. 19–22
18. J. Lienig, Electromigration-aware physical design of integrated circuits, in *18th International Conference on VLSI Design*, Jan 2005, pp. 77–82
19. 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
20. R. Martins, R. Póvoa, N. Lourenço and N. Horta, Current-flow & current-density-aware multi-objective optimization of analog ic placement. Integr. VLSI J. (in press, 2016). Reprinted from Integration, the VLSI Journal. With permission from Elsevier
21. R. Martins, N. Lourenco, N. Horta, Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates. Expert Syst. Appl. 42(23), 9137–9151 (2015). doi:10.1016/j.eswa.2015.08.020. Reprinted from Expert Syst. Appl. With permission from Elsevier

Chapter 6
Fully-Automatic Router

In this chapter, AIDA-L's fully-automatic Router is described; the inputs are the floorplan solution generated by the Placer, the netlist and the set of electric-current values for each terminal contained in the netlist. The solution space is then automatically explored, minimizing the total wiring area and complying with a set of electromigration (EM), IR-Drop and wiring symmetry (WS) constraints. The Router also ensures that each wire does not violate any of the design rules, nor shunts with other wires or shapes of the underneath cells. With all these validations and the huge search space, routing is extremely more complex and computationally more expensive than placement procedures, presented on Chaps. 4 and 5 of this book.

The first section of this chapter covers the general description of the Router architecture. Then, the routing procedure is explained, detailing each task implemented in AIDA-L's fully-automatic Router. The planning phases are described in sections "Electromigration-Aware Wiring Planner" and "Symmetry Planner", i.e., wiring and symmetry planners, the global routing procedure in Sections "Global Router: Step I—Multilayer Multiport Selection" and "Global Router: Step II—Steiner Point Assignment", and finally, in section "Detailed Router" the detailed routing phase is presented.

6.1 Router Architecture

AIDA-L performs a flexible fully-automatic routing with no manual setup required. The netlist and the set of electric-currents attached to each terminal, although obviously dependent from the circuit sizing considered, are provided independently from the floorplan and from all the processing tasks done in the Placer modules. That is, the topological relations between cells may change, the devices' sizes, and even the target technology design process may change, but the Router inputs are exactly the same in all situations, and do not require any additional setup or tuning.

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_6

6.1.1 Evolution of AIDA-L's Routing Paradigm

The actual fully-automatic implementation is a mature version of previous efforts to automate this task. First, a strict template-based approach where the designer, in a XML template file [1], provided for each wire the two exact terminal's pins to be connected, the geometric shape for the wire defined by a set of points, and also, the preferred conductor layer to be used. However, the need for all this information made the process of defining the template extremely time consuming. Additionally, the fixed position of the pins and the limited operators, which were intended to keep to a minimum the deviation from the designer definitions, introduced great limitations on the routing flexibility, strongly impairing the possible outcome. Moreover, any change in the topological relations between cells was likely to compromise all wire geometries, forcing complete template redesign. Although suited to abstract designer from technology details, the strict template-based approach was still cumbersome and did not yield physically achievable layouts for wide specification changes. The template-based routing was discontinued and further detail about the implementation can be found in [2].

Later, an approach where the designer defined the terminal-to-terminal connectivity, the symmetry constraints for those wires and, a set of sensitivity constraints for the nets, was used. These user-defined inputs were simpler to define than the initial routing setup, and were effective in guiding the evolutionary routing optimization. However, it was still too restrictive and was not able to accommodate for substantially different floorplans (topological relations between cells or devices' sizes), as the best set of terminal-to-terminal connections for each net for a given floorplan/set of devices' sizes was not adequate for a different one. Leading, once again, to time-consuming modifications to the terminal-to-terminal connectivity and all the corresponding constraints defined in the template file.

During the development of the current fully-automatic Router the possibility for the designer to manually define these connections was kept. However, it was removed from this dissertation since it is seldom used as it implies a great deal of work from the designer maintaining a template file. Also, at that time, the lack of electrical measures was compensated by extra constraints provided by the designer in the template file that affected the wires' width, which is not really needed in the actual routing design flow. For further information about this method please refer to [3].

6.1.2 Current Architecture/Design Flow

The proposed architecture/design flow is shown in Fig. 6.1, which depicts the main tasks performed by the fully-automatic Router. The process is executed in four main phases, denoted as Wiring Planner, Symmetry Planner, Global Router and Detailed Router.

The fully-automatic routing flow is briefly outlined in Algorithm 6.1, advancing the detailed description of the implementation of each processing block, which is remitted to the corresponding Sections.

The first block, the Wiring Planner (Sect. 6.2), performs the EM-aware Wire Planning. Starting from the netlist and the set of electric-current-sources (terminals with

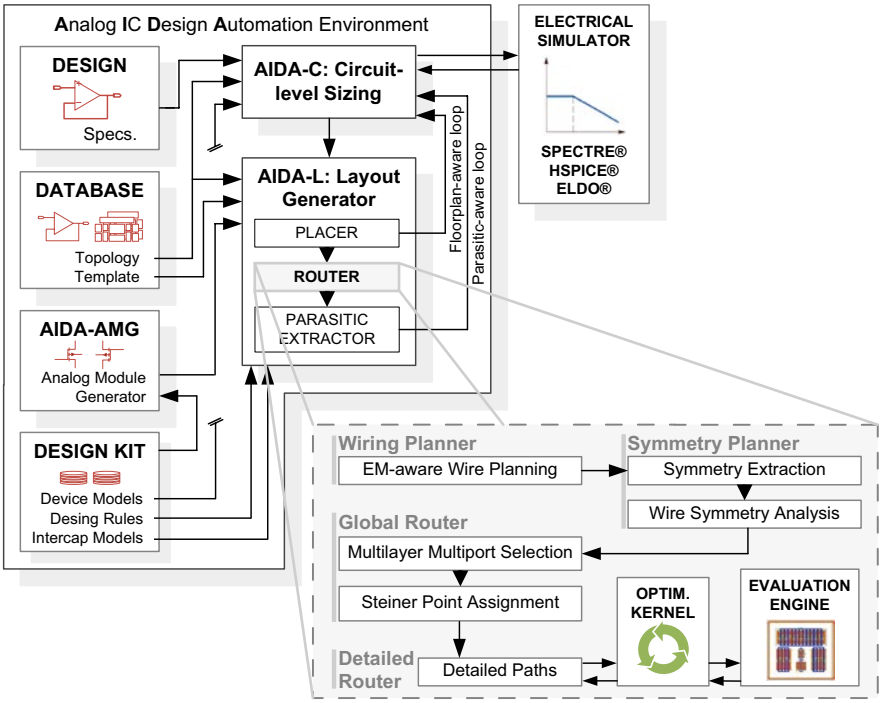


Fig. 6.1 Fully-automatic Router architecture: Wiring Planner, Symmetry Planner, Global Router and detailed Router blocks embedded in the AIDA-L's design flow

a positive electric-current value attached) and electric-current-sinks (terminals with a negative electric-current value attached), it derives the electric-current-correct wiring topology (WT) that connects all the terminals and provides the optimal terminal-to-terminal connectivity and current-flows, while minimizing the wiring area.

In the second block, the Symmetry Planner (Sect. 6.3), the information about symmetry between devices is extracted from the floorplan (Symmetry Extraction) and then for each terminal-to-terminal connection in the WTs, any symmetrical connection, if present, is identified (Wire Symmetry Analysis).

In the third block, the Global Router (Sects. 6.4 and 6.5), each terminal-to-terminal connection (or wire) is transformed into a rectilinear path where the ports that minimize the wire area, while implementing the detected symmetries, are selected from the multiple available ports of their corresponding terminals (Multilayer Multiport Selection). Afterwards, when the connectivity between all terminals is available, Steiner points are assigned to reduce congestion and wiring area by overlapping sections of adjacent rectilinear paths (Steiner Point Assignment).

The last step (Sect. 6.6) is the Detailed Router. At this stage, the expanded wires, whose width is determined according to their current-densities and subject to EM and IR-drop constraints, are instantiated in the layout, preserving WS and ensuring design rule and layout-versus-schematic correctness.

Algorithm 6.1: Fully-automatic Router Flow

input: **List<Net<List<Terminal<**Electric-current, **List<**Port**>>>>>** *Netlist* (each net constrains a list of multiport terminals, each one with an associated electric-current value and ports' locations)
output: **List<**Shapes**>** *DetailedRoutingPaths* (shapes of the detailed routing paths to be saved in GDSII format)

```
1.  //Electromigration-aware Wiring Topology (Section 6.2)
2.  for each net in the Netlist do
3.     List<Edge<Terminal, Terminal, w_i,j, I_i,j, c_i,j>> spanningTree:= create the electromigration-
       aware tree that connects all the terminals of the net¸ each terminal-to-terminal
       connection is an Edge instance
4.     List<Net<List<Edge>>> NetworksGraphs:= put the new spanningTree;
5.  //Symmetry Planner (Section 6.3)
6.  for each net i in the Netlist do
7.     for each Edge m in the NetworkGraph of net i do
8.        for each net j in the Netlist do
9.           for each Edge n in the NetworkGraph of net j do
10.             with m != n
11.             if(Edges m and n do not have a symmetric edge, evaluate symmetry information
                from the floorplan and search if a wiring symmetry heuristic matches) do
12.                set Edges k and l as a symmetric pair
13. //Global Router – Step I – Multilayer Multiport Selection (Section 6.4)
14. for each net i in the Netlist do
15.    List<Node> grid:= construct the multilayer multiport grid
16.       for each Edge m in the NetworkGraph of net i do
17.          if Edge m has a symmetric wire and is already routed do
18.             assign mirrored previous routing path
19.          else
20.             List<Net<List<Edge<Terminal, Terminal, w_i,j, I_i,j, c_i,j, List<Nodes>>>>>
                NetworksGraphs:= set field List<Node> with the shortest path from source
                Terminal to sink Terminal of Edge m using the enhanced A* algorithm
21. //Global Router – Step II – Steiner Point Assignment (Section 6.5)
22. for each net i in the Netlist do
23.    for each Edge m in the NetworkGraph of net i do
24.       for each Edge n in the NetworkGraph of net i do
25.          if a Steiner point assignment heuristic is verified do
26.             with m != n
27.             NetworksGraphs:= add a new Edge that corresponds to the overlapped area
                between Edges m and n, modify Edges m and n and search for the shortest path
                using the enhanced A* algorithm for the new source Terminal and/or sink
                Terminal
28.          if Edges m and n have symmetric wires do
29.             search in advanced heuristics for Steiner assignment and process
                NetworksGraphs
30. //Global Router (Section 6.6)
31. List<Shapes> DetailedRoutingPaths:= use single-net detailed optimization and multi-net
    detailed optimization over NetworksGraphs to obtain a DRC and LVS clean routing
    solution
```

6.2 Electromigration-Aware Wiring Planner

The problem in an EM-aware Router is to expand the wires' widths according to the current-densities imposed on them. Computing the rectilinear Steiner minimal tree without considering electric-currents and then widening the wires in the detailed routing will lead to very sub-optimal solutions in terms of total wire area and quality of the current-driven WT. Further, considering electric-currents only in later routing phases may result in improper electrical connections, which will lead to redesign cycles and iterations over the already routed nets. By considering current-densities in the wire planning, an EM-reliant routing may be easily achieved with much less effort.

The EM-aware Wiring Planner follows the steps introduced in Algorithm 6.2, and detailed throughout this Section. In the next sub-Section, reliable widths are derived for the interconnects. In Sect. 6.2.2 the EM-aware wire planning problem is formulated, after, the implemented solution is overviewed, and finally, in Sect. 6.2.4 the global procedure to ensure that a net is strongly connected (i.e., the net is represented by a single tree where all terminals are interconnected) is detailed.

Algorithm 6.2: EM-aware Wire Planning

	input: **List<Terminal<**Current, **List<**Port**>>** *Terminals* (list of multiport terminals in the net, with the associated current)
1.	**List<Edge<**Terminal, Terminal, $w_{i,j}$, $I_{i,j}$, $c_{i,j}$**>>** *NetworkGraph*:= create list of edges using the multilayer geometrical center of the terminals' ports, compute $w_{i,j}$ and determine $c_{i,j}$ *(Section 6.2.3.1)*
2.	*NetworkGraph*:= assign $I_{i,j}$'s using the minimum wire length approach
3.	**List<Edge<**Terminal, Terminal, $w_{i,j}$, $I_{i,j}$, $c_{i,j}$**>** *ResidualNetwork*:= create list of edges with the Residual Network from *NetworkGraph (Section 6.2.3.2)*
4.	**while**(infinite) **do** {
5.	*ResidualNetwork*:= use Bellman-Ford's algorithm to find a Negative cycle
6.	**if**(doesn't exist a Negative cycle in *ResidualNetwork*) **then**
7.	**break;**
8.	**else** {
9.	*NetworkGraph*:= update I_{ij}'s to remove the most negative cycle
10.	*ResidualNetwork*:= update the residual edges
11.	}
12.	} *(Section 6.2.3.3)*
13.	*//Ensure Strong Connectivity (Section 6.2.4)*
14.	**if**(the *NetworkGraph* is not strongly connected) **then**
15.	*NetworkGraph*:= intra-group edges are set with a $w_{ij} = 0$*NetworkGraph*:= add inter-group
16.	edges and compute w_{ij}
17.	**List<Edge>** *spanningTree* := compute Kruskal in *NetworkGraph*
18.	*NetworkGraph*:= add
19.	**return** *NetworkGraph*

6.2.1 Electromigration and IR-Drop-Reliable Interconnects' Widths

In order to obtain EM-reliable interconnect dimensions for analog integrated circuit (IC) design, realistic electric-current for each terminal needs be obtained. Process and temperature variations have a major importance in obtaining these electric-current values, however, not all the lithography process variations/issues can be properly analyzed in general approaches. In this approach, the electric-currents are automatically extracted during the circuit-level synthesis with AIDA-C using the electrical simulator. AIDA-C also considers corner cases, and particularly, the high temperature corners may be suited to obtain the set of worst-case electric-currents due to its importance in the median time to failure (MTF) of an interconnect, as introduced in Chap. 2 of this book.

The DC currents, where the metal is subject to an electron wind from a constant direction, are considered in all test cases when computing the interconnects' widths as their contribution to EM is proved to be larger than AC currents, and are easier to measure as they usually do not depend on the small-signal external stimulus. The AC currents effects due to the 'self-healing effect', i.e., a phenomenon of reversed material flows caused by alternating electric-current directions that results in a reduction on the overall material migration, their impact can be neglected for signals without extremely high peak-to-peak values [4]. Nevertheless, the routing approach is general enough to accommodate any current measurements the designer deem fit for any given circuits. The designer only needs to provide the proper testbench and stimulus.

For precise interconnect's width, the generalized Eq. (6.1) is used to compute the effective width w that satisfies the EM constraint for the root mean square current I_{eq} assigned to the interconnect and reference temperature T_{ref}, using the process-dependent parameters provided by the foundry, for state-of-the-art integration technologies [5–10]:

$$w_{EM} = \frac{I_{eq}}{J_{\max}\left(T_{ref}\right).h_{process}} \tag{6.1}$$

where, $J_{max}(T_{ref})$ and $h_{process}$ are constants tabulated for the technology design process. $J_{max}(T_{ref})$ is the maximum allowed current-density for the reference temperature, T_{ref}, used in the determination of I_{eq} or desired on-chip working temperature; and $h_{process}$ the nominal layer height. Summarizing, a higher temperature for the same current-density I_{eq} will require an increase in the interconnect's width in order to maintain the same MTF [4, 5].

Lets consider a general fabrication process, if the EM rule adjustment for temperature and product life time is guaranteed, it is assured that EM will have a negligible impact on the product reliability. For example, assuming a MTF of the product of 10 years and a reference temperature of 100 °C, the maximum allowable tabulated current-density J_{max}(100 °C) per unit width for *metal* 1 is 2.50 mA/μm.

If the assumed junction temperature is different from 100 °C, to maintain the same product life time of 10 years, $J_{max}(T_{ref})$ must be adjusted in according to Eq. (6.2):

$$J_{\max}\left(T_{\max}\right) = J_{\max}\left(100\text{"}C\right).e^{\left(\frac{6000}{T_{\max}\left(\text{"}K\right)}-15\right)} \tag{6.2}$$

However, while the EM constraint may be fulfilled, due to the inherent resistance of the interconnects undesirable voltage drops across the networks are still possible to occur. The inequality in Eq. (6.3) determines the maximum assignable length l_{max} for a interconnect (source to sink length), in respect to a fixed maximum tolerable IR-drop voltage V_{max}, which provides a safe length margin to reduce voltage drops, where $r_0(T_{ref})$ is the conductor sheet resistance [10].

$$l_{eff} \le l_{\max} \sim \frac{V_{\max}.w_{EM}}{I_{eq}.r_0\left(T_{ref}\right)} \tag{6.3}$$

This way, to reduce the IR-drop effect is required that the length of the IR-drop-sensitive lines is kept as short as possible, this is ensured both in the implemented wire planning as in the pathfinding algorithm, that searches for the shortest paths available. However, if the effective length l_{eff} of the assigned path still surpasses l_{max}, the wire must be re-widened according to Eq. (6.4) to decrease the resistance of the conductor per unit square.

$$w_{IRdrop} = \frac{I_{eq}.l_{eff}.r_0\left(T_{ref}\right)}{V_{\max}} \tag{6.4}$$

When both w_{EM} and w_{IRdrop} values lead to an effective wire width w_{eff} below the minimum width manufacturable, $w_{min\_process}$, this value is used instead, as summarized in Eq. (6.5):

$$w_{eff} = \max\left\{w_{EM}, w_{IRdrop}, w_{\min\_process}\right\} \tag{6.5}$$

6.2.2 Problem Formulation

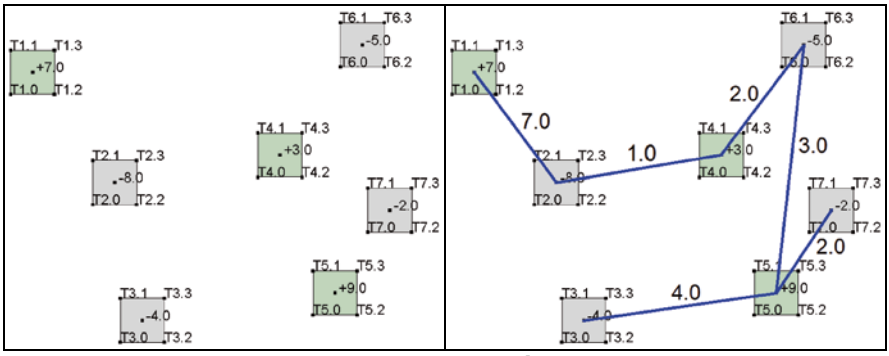
Given a set of k terminals $\{T_1, T_2,\ldots, T_k\}$ of a signal net, where m terminals are considered as electric-current-sources, henceforward indicated by the positive electric-current value attached to the terminal, and n terminals considered as electric-current-sinks, in opposition, indicated by the negative electric-current value, construct the tree that minimizes the total wire area of the net:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} l_{i,j} \times I_{i,j} \tag{6.6}$$

where the wire length $l_{i,j}$ can be computed by Eq. (6.7) and $I_{i,j}$ is the electric-current in the wire that connects T_i to T_j, due to its proportionality, the electric-current is used instead of the actual wire width. Further, the topology must be constructed while satisfying the Kirchhoff's current laws in every terminal. Summarizing, the idea of the wiring planning step is to find the set of terminal-to-terminal connections for a set of unconnected terminals, in a way that the number of long current-intensive connections is reduced as much as possible.

$$l_i = \left|x_i - x_j\right| + \left|y_i - y_j\right| + \left|z_i - z_j\right| \times V_{cost} \tag{6.7}$$

Retrieving the example introduced in [7], a set of seven terminals are considered $\{T_1, T_2,\ldots, T_7\}$ as being elements of the same net, defining three terminals as electric-current-sources $\{T_1, T_4, T_5\}$ and four as electric-current-sinks $\{T_2, T_3, T_6, T_7\}$. The attached positive/negative electric-current values $\{I_1, I_2,\ldots, I_7\}$ are normalized as: T_1 (+7), T_2 (−8), T_3 (−4), T_4 (+3), T_5 (+9), T_6 (−5) and T_7 (−2), which are marked at the center of the multiport terminals, as presented in Fig. 6.2a. Each of the terminals was transformed into a four port quadrangular terminal geometry for the demonstration of the methods of this Section, but any multilayer multiport geometry could be defined.



a **netlist**

T_1 @ +7 [mA]
T_2 @ –8 [mA]
T_3 @ –4 [mA]
T_4 @ +3 [mA]
T_5 @ +9 [mA]
T_6 @ –5 [mA]
T_7 @ –2 [mA]

b **EM-aware wiring topology**

$T_1 \rightarrow T_2$ @ 7 [mA]
$T_4 \rightarrow T_2$ @ 1 [mA]
$T_4 \rightarrow T_6$ @ 2 [mA]
$T_5 \rightarrow T_3$ @ 4 [mA]
$T_5 \rightarrow T_6$ @ 3 [mA]
$T_5 \rightarrow T_7$ @ 2 [mA]

Fig. 6.2 (**a**) Topology with multiport terminals, netlist and attached electric-current vectors, T_1(+7), T_2(−8), T_3(−4), T_4(+3), T_5(+9), T_6(−5) and T_7(−2), which are marked at the geometrical center of the multilayer terminals' ports [9]. The number after the terminal's designation is the port's index. (**b**) EM-aware WT, i.e., terminal-to-terminal connectivity and current-densities for each connection [11, 25]

6.2.3 *Optimal Wire Planning*

From the EM-aware wire planning approaches found in the state-of-the-art overviewed in Chap. 2 of this book, WiT [11] proved to outperform the previous implementations both in terms of minimizing the total wiring area, as in the computational time required to achieve the solutions, for any range of single-port terminals. The approach is based on the proof of the greedy-choice property, i.e., only electric-current from sources to sinks is considered, source-to-source flow or sink-to-sink flow is not supported. Due to this property, the electric-current assignment problem can be strongly simplified and dealt as a class P problem instead of NP-hard. The terminal-to-terminal connectivity and flows of the optimal WT are determined in WiT using the multilayer geometrical center of the terminals' ports, Fig. 6.2b. WiT relies on three steps, network graph construction, residual network construction and negative cycle cancelling, in Sects. 6.2.3.1, 6.2.3.2 and 6.2.3.3, respectively.

Also, during the Wiring Planner procedure, the concept of power net is introduced. Again, the flow of electric-current-sources is only redirected to the electric-current-sinks. Since *vdd* or *vss* nets have only current-sources or current-sinks, it is impossible to construct the proper WT. So, a power net is defined as a conductor stripe that can be placed above, on the left, on the right or below the circuit, adjusted to the width or height of the circuit, and it is instantiated automatically similarly to what it is done in digital layout design. It contains a new multiport terminal which is added to the respective net, with the same name of the net, and an electric-current associated which is the negative value of the sum of all terminals' electric-currents.

6.2.3.1 Network Graph

The flow network graph for the problem is constructed by assigning a directed edge $e_{i,j}$ from the multilayer geometrical center of the terminals' ports of each of the sources T_i, to the geometrical center of the terminals' ports of each of the sinks T_j. Each of the edges of the network graph, Fig. 6.3a, has three associated values ($l_{i,j}$, $I_{i,j}$, $c_{i,j}$), namely: the wire length $l_{i,j}$, computed by Eq. (6.7); the electric-current $I_{i,j}$, which are the parameters to be found; and the current capacity $c_{i,j}$, which corresponds to the maximum electric-current value that can be assigned for the edge $e_{i,j}$:

$$c_{i,j} = \min\left(\left|I_i\right|, \left|I_j\right|\right) \tag{6.8}$$

The first step is to assign an initial solution for the flow of electric-currents $I_{i,j}$. The method used is to select the edges with the minimum wire length first and greedily assign the maximum electric-current possible to that edge, while not surpassing the maximum electric-current required on a sink or withdrawing more electric-current than the currently available on a source. The result of the minimum wire length initialization for the proposed flow network is presented in Fig. 6.3b, edges with zero electric-current are removed from the network.

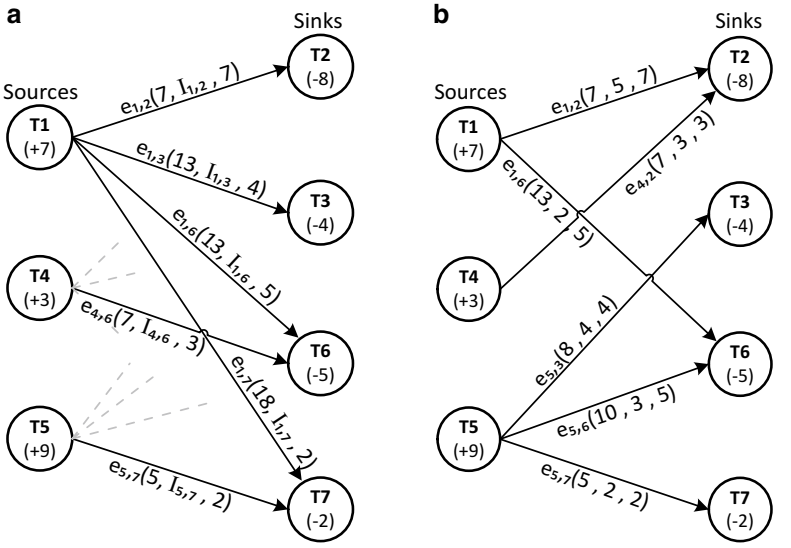


Fig. 6.3 (**a**) Flow network graph for the example of Fig. 6.3, $e_{i,j}(l_{i,j}, I_{i,j}, c_{i,j})$. Some edges were omitted for clarity. (**b**) Electric-currents $I_{i,j}$ assigned for the minimum wire length greedy initialization, edges with $I_{i,j}=0$ are removed from the flow network [25]

6.2.3.2 Residual Network

The construction of the residual network is based on two principles: for any edge $e_{i,j}$, it is not possible to inject more electric-current than $(c_{i,j}-I_{i,j})$, i.e., it cannot be injected a quantity of flow that surpasses its capacity; and, if an edge $e_{i,j}$ has an electric-current I_{ij} different from zero assigned, then we can remove at most $I_{i,j}$ electric-current from that edge. This way, the residual network will have forward edges, for increments of electric-current, and backward edges, for decrements of electric-current. Concretizing these two principles, a forward residual edge $re_{i,j}$ of the residual graph has a residual capacity $rc_{i,j}$ of:

$$rc_{i,j} = c_{i,j} - I_{i,j} \tag{6.9}$$

and backward residual edge $re_{j,i}$ has a residual capacity $rc_{j,i}$ of:

$$rc_{j,i} = -I_{i,j} \tag{6.10}$$

with a residual negative wire length $rw_{j,i}$ of:

$$rw_{j,i} = -w_{i,j} \tag{6.11}$$

If a forward/backward edge has a residual capacity equal to zero it is removed from the residual network. The residual network for the set of electric-currents $I_{i,j}$ determined with the minimum wire length greedy initialization can be constructed by applying these rules to each of the edges $e_{i,j}$, and is presented on Fig. 6.4a.

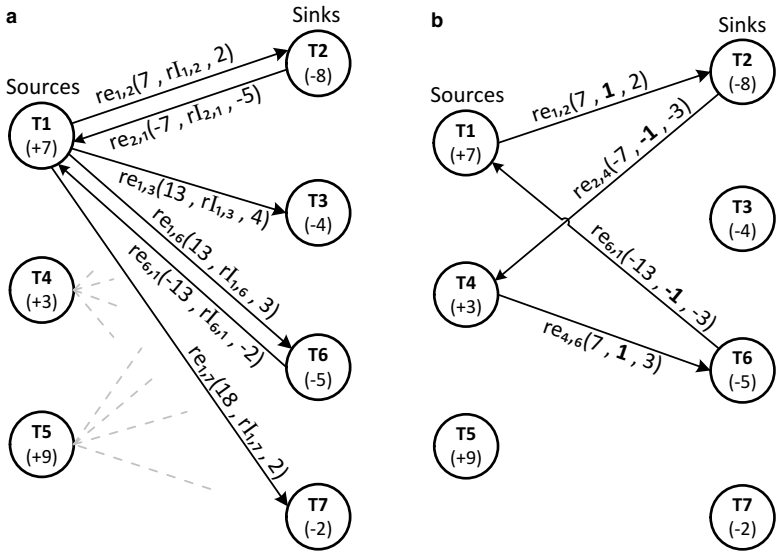


Fig. 6.4 (**a**) Residual network for the flow assigned in Fig. 6.4, some edges were omitted for simplicity. (**b**) Negative cycle detected, wire length $= rw_{4,6} + rw_{6,1} + rw_{1,2} + rw_{2,4} = 7 + (-13) + 7 + (-7) = -6$ [25]

6.2.3.3 Negative Cycle Cancelling

The negative cycle problem consists on finding or proving that a cycle with negative wire length within a network does not exist [12]. In this approach, the Bellman-Ford algorithm is used to find a negative cycle on the residual directed graph. If a cycle with a negative wire length exists, then, the electric-current assigned for the actual flow network graph is not optimal, and can be minimized by changing the electric-currents affected by the negative cycle. Consider the residual network of Fig. 6.4a, the cycle with most negative wire length found is presented in Fig. 6.4b. The maximum allowed electric-current, I_{max}, that can be injected in the detected cycle corresponds to the smaller absolute value among all capacities of residual edges of the negative cycle, which for the current example is:

$$I_{\max} = \min\left\{\left|rc_{4,6}\right|, \left|rc_{6,1}\right|, \left|rc_{1,2}\right|, \left|rc_{2,4}\right|\right\} = 2 \tag{6.12}$$

This way, a two-unit flow is injected in the negative cycle of Fig. 6.4b, i.e., by removing a two-unit flow from residual edges re_{61} and re_{24}, and injecting that flow on re_{12} and re_{46}, the total wire length is reduced. The flow network is updated, and also, the residual graph to reflect the changes.

The optimization is done by executing these steps iteratively until no negative cycles exist in the residual network, as presented in Algorithm 6.2. The negative cycle detection algorithm ensures that the solution is always improved until the optimal

solution, in the conditions of the problem, is found. For the current example, since no more negative cycles were detected, the optimal solution is found and was already presented in Fig. 6.2b.

6.2.4 Strongly Connected Network

In a signal net it is required that all the terminals are interconnected. However, by the use of an algorithm that assigns the electric-current of all electric-current-sources directly to the electric-current-sinks [11, 13] it is possible to obtain electric-current-correct sub-nets inside the same net. Furthermore, in real IC design cases, the presence of zero or approximately zero-current terminals cannot be interpreted as current-sources or sinks, and therefore, these terminals will not be included with the remaining in the EM-aware WT. The direct problem from this is that the whole net became a forest (i.e., multiple unconnected EM-correct trees for the same net), and consequently, unconnected, and it is required to add 'dummy' wires between trees to ensure that the tree is strongly connected while simultaneously minimizing the number of further connections.

For small nets and number of trees, the problem could be solved by an exhaustive search among the unconnected terminals; however, for large nets and number of sub-nets, the exhaustive search for the best terminals to establish the connectivity is problematic. The use of multiple electric-current vectors for the terminals may conceal the problem, but only in the cases where local 'three-point Steinerization' techniques are used [7–9]. Consider the terminal topology of Fig. 6.2 with a new set of current values: $T_1(+7)$, $T_2(-8)$, $T_3(-4)$, $T_4(+3)$, $T_5(+6)$, $T_6(-2)$ and $T_7(-2)$, on which the optimal WT obtained by negative cycle detection is presented in Fig. 6.5a. In this topology there are two unconnected sub-nets, namely, the $\{T_1, T_2, T_4, T_6\}$ and the $\{T_3, T_5, T_7\}$ net.

In our approach, a global procedure based on a minimum spanning tree (MST) construction is used to ensure the strong connectivity of the net, while simultaneously minimizes the number of required connections. The edges that connect terminals of the same sub-net, intra-group terminals, are assigned with a wire length of zero since they are strongly connected. Then, inter-group edges weighted with the distance between them are created from each terminal of a sub-net to each other terminal of all remaining sub-nets. Kruskal's algorithm [14] is then used to compute the MST containing all the terminals. The non-zero edges of the MST are then added to the optimized network graph obtained in the previous step, as depicted in Fig. 6.5b. The DC current in these wires is zero and they will be expanded with minimum width allowed, $w_{min\_process}$, according to Eq. (6.5).

After performing this procedure, for each net of the circuit, a set of terminal-to-terminal connection topologies (each one for each net) are obtained, which can now be dealt by a traditional point-to-point pathfinding algorithm.

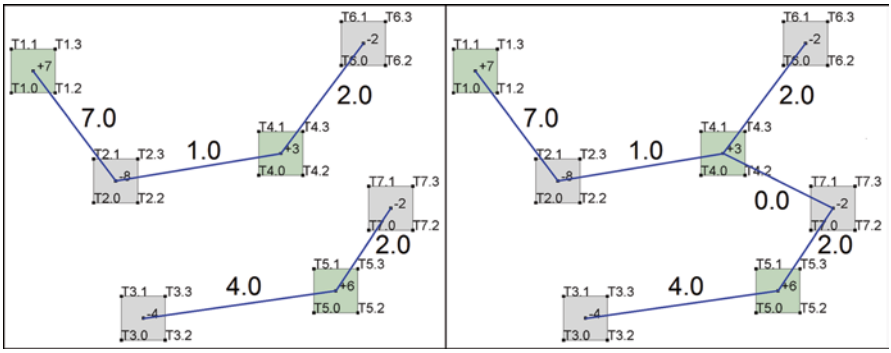


Fig. 6.5 (**a**) EM-aware wire planning, for the current-densities: $T_1(+7)$, $T_2(-8)$, $T_3(-4)$, $T_4(+3)$, $T_5(+6)$, $T_6(-2)$ and $T_7(-2)$. Two electric-current-correct sub-nets, $\{T_1, T_2, T_4, T_6\}$ and $\{T_3, T_5, T_7\}$, are created inside the same net. (**b**) WT after the global procedure to ensure strong connectivity. Edges with zero current-density have the width set to $w_{min\_process}$ [25]

6.3 Symmetry Planner

In the traditional analog design flow, the routing task is usually triggered when the placement task is complete. While WS is intrinsically related with the devices' symmetry, a symmetrical placement is not enough to ensure WS during the routing process. It is mandatory to plan WS and implement it during both global and detailed routing phases [15]. Further, an asymmetric placement will certainly lead to an asymmetric routing, since symmetric wires are only feasible if the multiport (MP) terminals of the paths being routed are symmetric with respect to a symmetry axis.

6.3.1 Symmetry Extraction

Instead of routing, previous efforts heavily focused on constraint-driven placement, with an extensive use of layout constraints [16]. So, in order to generate the symmetrical routing for the provided template/floorplan, valuable information of the relative positioning between devices/sub-cells is already available in most analog design automation frameworks and only needs to be properly extracted. In the case of the proposed template-based Placer, matching and symmetry constraints are included in the high level floorplan guidelines of the hierarchical template file used to generate the floorplan.

In Fig. 6.6, a simple partition of a layout template with a vertical symmetry axis (marked with the dashed line) is presented and the correspondent relations between cells identified. By analyzing a partial or fully symmetric template/floorplan, two relations can be identified for the cells: symmetry pairs (SPs), if two devices/sub-cells are matched (i.e., geometrically identical), mirrored and placed symmetrically in respect to a symmetry axis; and self-symmetric (SS) cells, if devices/sub-cells are

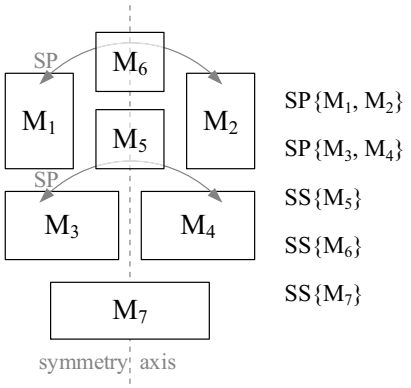


Fig. 6.6 Symmetry pairs (SPs) and self-symmetric (SS) cells identified for the simple template/floorplan

placed centered with respect to the symmetry axis. These considerations can also be used at higher levels of the floorplan hierarchy, for intra- and/or inter-floorplan cell symmetry identification.

6.3.2 Wire Symmetry Analysis

When the set of symmetry relations between cells and consequently between cells' terminals is available, symmetries between wires can be identified. Each terminal-to-terminal connection from each EM-aware WT (one for each net in the netlist) is tested with all other connections from the same and other WTs. Each connection only relates two modules (the ones being connected), independently from the number of blocks present in the floorplan. Three different heuristics are used for wire symmetry pattern identification that relate two terminal-to-terminal connections at a time.

Considering a generic situation of seven terminals defined by three SPs and a SS cell illustrated on Fig. 6.7a. The first heuristic considered defines that two terminal-to-terminal connections, C_1 and C_2, can be deemed symmetric, $S\{C_1, C_2\}$, if condition (6.13) is true. Here, a symmetric relation is established between two connections on opposite sides of the symmetry axis, as illustrated in Fig. 6.7b, c.

$$C_1\left\{M_i, M_p\right\} \& C_2\left\{M_j, M_q\right\} \& SP\left\{M_i, M_j\right\} \& SP\left\{M_p, M_q\right\} \tag{6.13}$$

The second heuristic, which also considers a situation of four terminals, defines that two symmetric connections may cross the symmetry axis if satisfy the condition (6.14), i.e., a 'connector' wiring structure [17]. This can be depicted in Fig. 6.7d.

$$C_1\left\{M_j, M_p\right\} \& C_2\left\{M_i, M_q\right\} \& SP\left\{M_i, M_j\right\} \& SP\left\{M_p, M_q\right\} \tag{6.14}$$

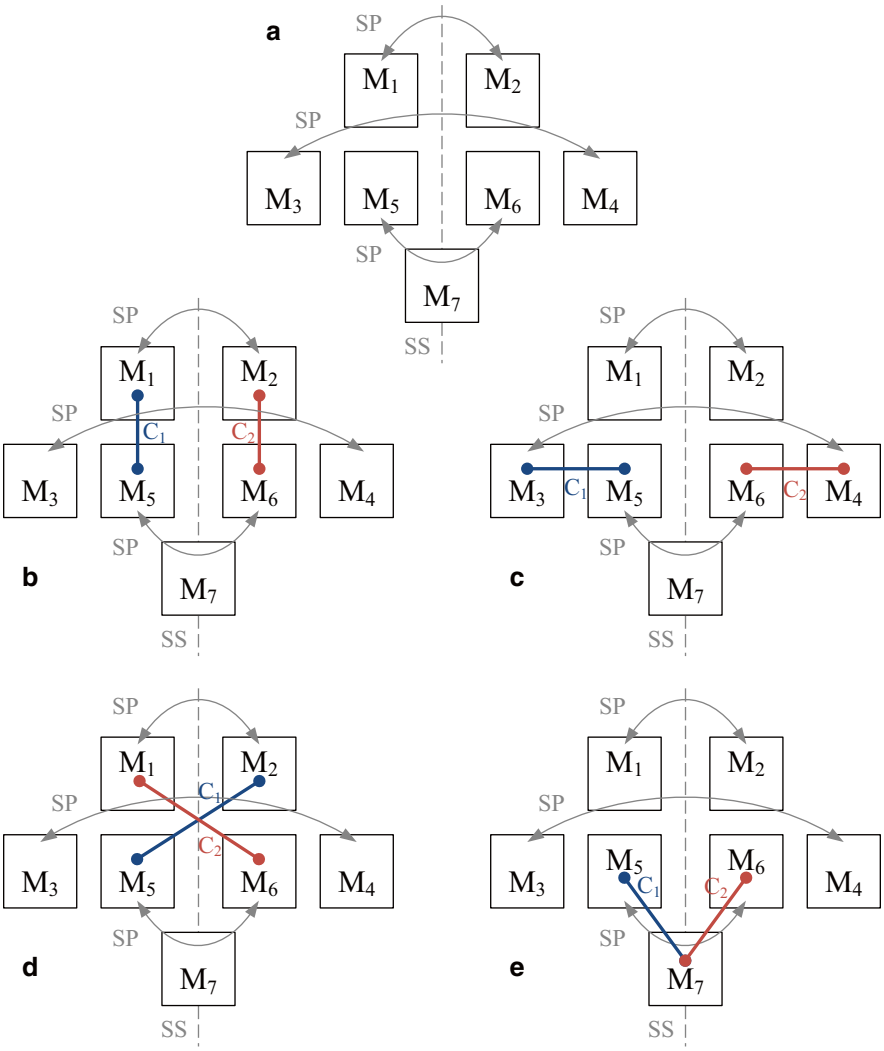


Fig. 6.7 Heuristics for the identification of WS over the genetic terminal disposition of (**a**): (**b**, **c**) Symmetric in relation to the symmetry axis; (**d**) Crossing the symmetry axis; and (**e**) with a SS cell

In this case, the two symmetric wires overlap, and, if mirrored directly it will result in a short-circuit (if C_1 and C_2 are from different nets). To obtain a good matching of the parasitic without introducing any error the overlapped sections of both connections are placed over the symmetry axis but in different layers.

Finally, for each two terminal-to-terminal wires that connect a SP to the same SS terminal, a third heuristic defines that if condition (6.15) is satisfied a new $S\{C_1, C_2\}$ can be assigned. This is illustrated in Fig. 6.7e.

$$C_1\left\{M_i,M_p\right\}\&\,C_2\left\{M_j,M_p\right\}\&\,SP\left\{M_i,M_j\right\}\&\,SS\left\{M_p\right\} \tag{6.15}$$

6.4 Global Router: Step I—Multilayer Multiport Selection

For the Router to efficiently connect all terminals of the nets, without the intervention of the designer, it must be aware of all possible locations and manufacturing layers where the MPs of a terminal might be, which doesn't happen when using equivalent 'dot-models'. Starting from the terminal-to-terminal connectivity obtained during wire planning, the problem of routing multiport multiterminal (MP/MT) signal nets is sketched in the next sub-section, afterwards, the multilayer grid construction with obstacle-avoidance is detailed, and finally, the MPs that provide the shortest paths are selected.

6.4.1 Multiport Multiterminal Signal Nets

The problem of routing MP/MT signal nets is addressed in the literature as the group Steiner problem [18]. In the presence of groups of ports, each group corresponding to a different terminal, the group Steiner problem can be formulated as follows: given an undirected weighted graph $G=(V, E, cost)$ and a set of k different terminals $T=\{T_1,\ldots, T_k\}$ with the ports of $T_i \subseteq V$, find the minimum-cost tree in G containing at least one port from each terminal T_i. By the use of a collection of ports for each terminal instead of a single-port, and also, considering multilayers and obstacles, the complexity of the original NP-complete Steiner problem is further increased.

There are two different approaches to this problem: (1) the weak-connectivity version, where only one port of each terminal is routed and all the connections are made to generate a single tree. A set of Steiner points may be used to reduce the total interconnect length, as depicted in Fig. 6.8a; and (2) the strong-connectivity version, which allows the use of intra-group connections along with the use of optional

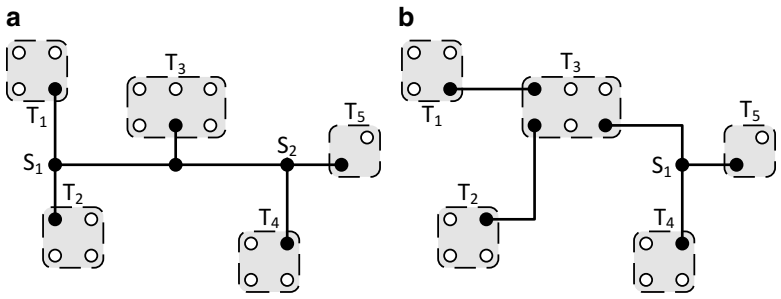


Fig. 6.8 MP/MT nets, each set of points within the same boundary represent the available ports of a terminal T_i, a point out of any boundary represents a Steiner S_j; (**a**) weak-connectivity and (**b**) strong-connectivity solution [19]

Steiner points, assuming that all ports of a group are connected internally, as depicted in Fig. 6.8b. In the last approach, the result is a forest of trees instead of a single support tree. Approximations for the weak-connectivity version of the problem, Fig. 6.8a, can be found by the use of the algorithms described in [19].

In electronic design automation, while single-port cells may be suitable in digital architectures or at system-levels of analog-or-mixed signal (AMS) circuits, where the terminals of the macro-cells are usually located at the higher metal layers of those available in the manufacturing process, at analog cell-level they degrade the quality of the solutions. These single-port MT approaches usually require a previous pin-assignment estimation procedure (by solving an instance of the weak-connectivity version of the group Steiner problem) or the inclusion of expert knowledge in the setup process, which lead to higher setup times and/or sub-optimal solutions. At cell-level, since most of the device's ports are located in the lowest layers, a complicated terminal geometry can easily have tens of ports located on multiple fabrication layers, and the use of an automatic MP approach greatly increases the flexibility of the devices' placement in any orientation.

In the proposed approach, the connection between equivalent ports is ensured in the generation of the module, which allows addressing the proposed Router as a strong-connectivity group Steiner-based problem.

6.4.2 Multilayer Multiport Obstacle-Aware Grid

Before selecting the appropriate ports for the terminal-to-terminal connectivity attained in the EM-aware wire planning, it is necessary to construct a multilayer rectilinear grid for the problem, which consists on an undirected graph considering three-dimensional vertices. The device's structures are ignored during the grid construction, only the port locations and obstacles locations (which may be located inside device's structures) are of concern. The generic terminal T_1 consisting of four distinct ports all located in the same manufacturing layer, will be used to illustrate the implemented methods, as depicted in Fig. 6.9a. Assume that all ports $\{T_{1.0}, T_{1.1}, T_{1.2}, T_{1.3}\}$ are electrically-equivalent locations for the same terminal, e.g., a transistor drain.

Initially, each of the terminal's ports is extended to each of the fabrication layers considered, which are presented in Fig. 6.9b, vias are used to represent the connection between of two vertices in different manufacturing layers. If an obstacle blocks the extension of a port to another layer, that vertex is not instantiated, nor the resulting extensions to the layers above or below that one, as depicted in port $T_{1.3}$. The same method is used to extend the corners of each obstacle to each different layer, and instantiate the subsequent possible vertices in locations not blocked by other obstacles, Fig. 6.9c.

The process of extending ports and obstacles' corners is performed simultaneously, and the result is the main set of available vertices in each layer, which is presented in Fig. 6.9d. After having all vertices for each different layer assigned, a

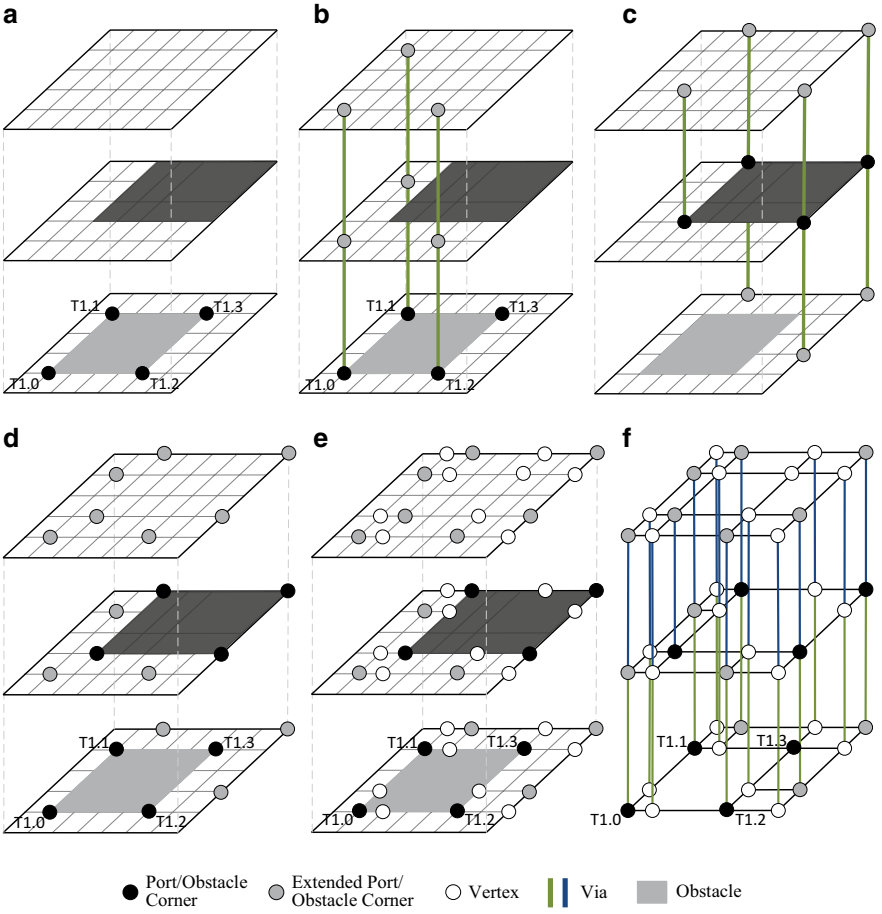


Fig. 6.9 (**a**) Example with one MP and two obstacles (one of them located inside the MP structure). Three fabrication layers are considered. (**b**) Each of the ports from a terminal is extended to each layer. (**c**) Each corner of an obstacle is extended to each layer. (**d**) Result of the processing of step (**b**) and (**c**). (**e**) For each layer, horizontal and vertical lines are passed through each vertex, which originate the remaining vertices of the grid. (**f**) The assignment of rectilinear neighbors to each vertex results in a custom grid for the current ports and obstacles [25]

vertical and horizontal line is passed through each vertex, i.e., through ports, extended ports, obstacles' corners and extended obstacles' corners. The intersections of those lines define the remaining vertices of the directed graph, Fig. 6.9e.

Finally, for each vertex the rectilinear neighbors are assigned, i.e., the directed edges of the graph, with a cost provided by Eq. (6.7). Each vertex may have a maximum of six neighbors, i.e., a neighbor at left, right, above or below in the same manufacturing layer, and neighbor in the exact (x, y) coordinates but located one layer above (x, y, $z+1$) or one layer below (x, y, $z-1$). For each neighbor assigned in a manufacturing layer above or below, a via is required. When all the possible neighbors are assigned, the final custom grid is achieved, Fig. 6.9f.

6.4.3 Multiport Selection

The optimal path for a given terminal-to-terminal connection is highly dependent of the obstacles present in the floorplan. To select the best ports to be connected and the corresponding path, a variation of the A* search [20], operating over the sparse non-uniform multilayer grid presented in the previous section, where all possible routes from all start ports to all end ports are explored simultaneously, is proposed. By expanding all the routes simultaneously, the number of nodes explored is reduced, as only the best route is completely explored. Hence, All MPs from each terminal are explored efficiently to achieve the shortest path in the presence of obstacles and increase the feasibility of WS. The modified A* search pseudo code is presented in Algorithm 6.3, where the cost function for leaf l while searching the path to terminal port j is given is given by Eq. (6.16):

$$ct\left(l,Te_j\right) = pathCost\left(l,Te_j\right) + dist\left(l,Te_j\right) \tag{6.16}$$

where the *pathCost*(l, *Tej*) is the real path size from the start terminal port to node l found so far in the way to the end terminal port node Te_j, and *dist*(l, Te_j) is the Manhattan distance between the leaf node l and the end terminal port node Te_j.

Algorithm 6.3: Multiport Selection

```
input:  Terminal A, Terminal B, List<Nodes> grid
1.  Queue queue := empty //empty set ordered by cost
2.  for(Port pa in A) do
3.     for(Port pb in B) do                  // add one entry for the
4.        queue.add(new QueueEntry(          // path between each
5.           grid.getNodeLocatedAt(pa),     // start port to each
6.           grid.getNodeLocatedAt(pb)))    // end port
7.     for(Node n in grid) do
8.        n.pathCostOnTheWayTo(             //initialize node cost
9.           grid.getNodeLocatedAt(pb)) := POS_INFINITE
10. while(queue.isNotEmpty()) do
11.    node u := queue.remove()
12.    if(u is endNode) return backtrackPathTo(u)
13.    for(Node v in n.neighbours) do
14.       tentativeCost = u.pathCost + dist(u,v)
15.       if(v.pathOnTheWayTo(u.end) > tentativeCost)
16.          v.pathOnTheWayTo(u.end) := tentativeCost
17.          queue.add(new QueueEntry(v,u.end))  //pass the end port
                                                 //to evaluate the cost
18. return null //no path found
```

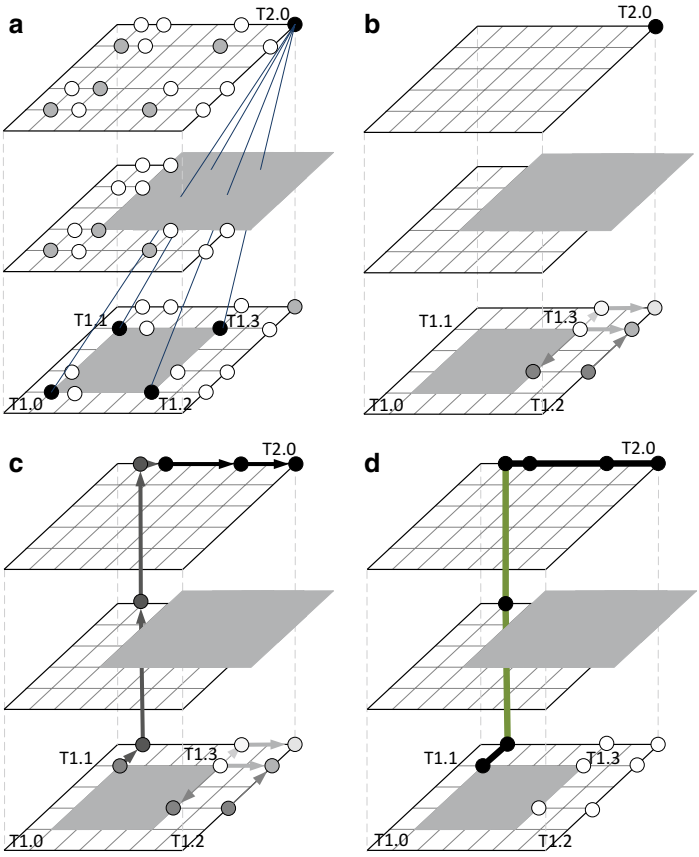


Fig. 6.10 Example of the parallel A* search: (**a**) Grid for a 4 to 1 connection example; (**b**) A* expansion of nodes from T1.3 (ordered from *lighter* to *darker*); (**c**) A* expansion of nodes from T1.1 (ordered from *lighter* to *darker*); (**d**) Nodes in the path (*darker*) and explored nodes not in the shorter path (*lighter*) [25]

Figure 6.10 illustrates the execution of the shortest path search between the generic terminal T_1, with four ports, and another generic terminal T_2, with only one port (for illustration purposes). Where Fig. 6.10a shows the grid, the vertices and the slant edges mark the four possible connections between T_1 and T_2. Figure 6.10b shows the initial expansion of nodes during the search. First, node $T_{1.3}$ is expanded because the $dist(T_{1.3}, T_{2.0})$ is the smallest, but the path is blocked, and while going around the obstacle, $T_{1.1}$ becomes the node closer to $T_{2.0}$ and starts to be explored, leading to the final path, Fig. 6.10c. $T_{1.0}$ and $T_{1.2}$ were not explored. Figure 6.10d highlights the selected path connecting $T_{1.1}$ to $T_{2.0}$. The efficiency of this approach can be seen in this simple example, where only 13 nodes were explored, and some of the ports of T_1, were not even explored. In networks this small, it is really of no importance, but for larger networks with many ports per terminal the savings are relevant.

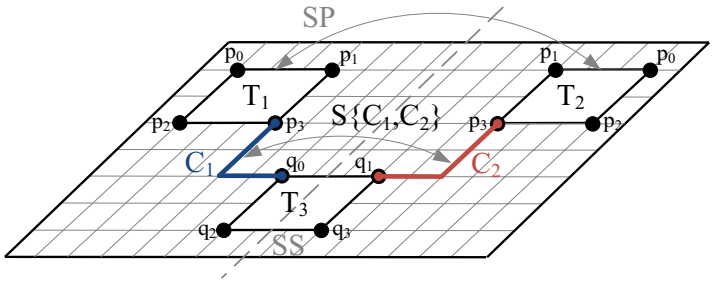


Fig. 6.11 Heuristic of Fig. 6.7c applied to three 4-port terminal structures

The MP selection is executed for all edges in the MST obtained from the EM aware wire planning to provide the port-to-port optimal rectilinear forest. By considering multilayer and obstacle-avoidance a great reduction on the required wiring length is achieved while simultaneously assigning the rectilinear paths, which are obtained from the multilayer multiport rectilinear obstacle-aware grid.

6.4.4 Wiring Symmetry in the Pathfinding Algorithm

Considering MPs is fundamental to achieve symmetry in the wires. In a pair of symmetric wires, S{C_1, C_2}, only one requires the execution of the path-finding algorithm and the other is mirrored. In the example of Fig. 6.11 the heuristic of Fig. 6.7c is implemented considering four-port MP structures, showing path C_2 as the mirrored path C_1. It is important to notice that terminal T_3 has two ports that are routed simultaneously for two different symmetric connections. When mirroring paths there are two distinct situations in the selection of MPs:

- If the terminal is part of a SP (i.e., cells mirrored in relation to the symmetry axis) the selected port is the same, e.g., port p_3 of mirrored terminals T_1 and T_2.
- For SS terminals/cells, the port selected inside the same terminal is the one located symmetrically in relation to the symmetry axis, e.g., ports q_0 and q_1 of terminal T_3.

Although only four port quadrangular terminal structures in the same fabrication layer are represented in the example, the same principles are used for any number of ports distributed by different fabrications layers. While WS is usually ensured in the routing within devices of the Module Generator leading to symmetric port locations, when the module is not fully symmetric the port closest to the symmetric location can also be used instead. In this case the wire symmetry is not exactly meet, but the corrections are made always maximizing the symmetric area as possible.

6.5 Global Router: Step II—Steiner Point Assignment

With the port-to-port optimal rectilinear path found for all connections of a given network, Steiner points that will reduce the total wire area and wiring congestion are assigned preserving the currents imposed on each path and WS.

6.5.1 Basic Assignment

With the port-to-port optimal rectilinear forest done, it is possible to assign some Steiner points that will reduce the total wire area under the EM considerations. The Steiner points are assigned for the current paths by considering two directed edges (i.e., an edge connects a 'source' port to a 'sink' port) at a time. If the edges start on the same source, or, end on the same sink, it may be suited to have their paths partially overlapped and a Steiner point introduced. Some heuristics are used to determine the optimal Steiner point position for the current grid, which maximize the edges' overlap. In Figs. 6.12 and 6.13 are presented a set of the possible combinations for the assignment of Steiner points.

There are two distinct situations: if a common source connects two different sinks (ports or previously assigned Steiner points) located in the same quadrant or adjacent quadrants; or if two different sources (ports or previously assigned Steiner points) located in the same quadrant or adjacent quadrants connect the same destination. If the requirements are satisfied, then, the two edges are divided into three different edges where one of the new edges current correspond to the sum of the two currents I_i and I_j, and a Steiner point is created at their junction. This proceeding can be used recursively to use the new paths of the previously assigned Steiner points.

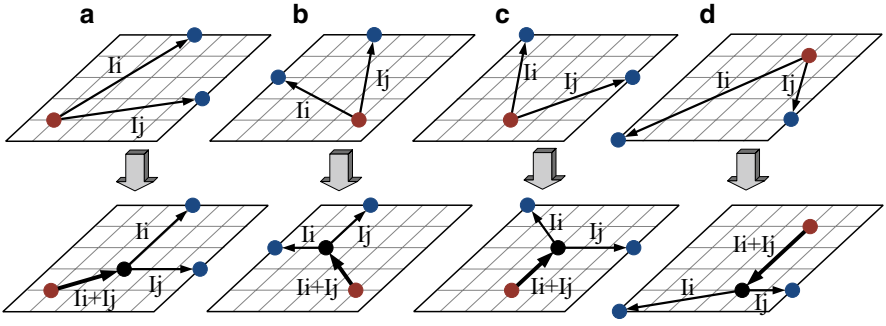


Fig. 6.12 Two edges with common source and different sinks: (**a**, **b**) the two sinks are in the same quadrant (two of four possible combinations represented), (**c**, **d**) the two sinks are in adjacent quadrants (two of four possible combinations represented) [25]

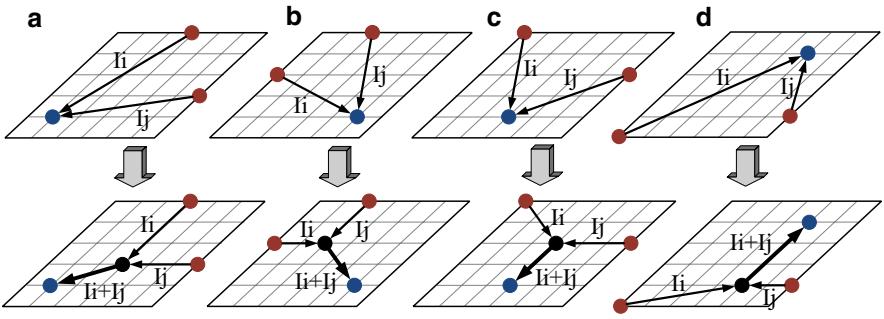


Fig. 6.13 Two edges with different sources and common sink: (**a**, **b**) the two sources are in the same quadrant (two of four possible combinations represented), (**c**, **d**) the two sources are in adjacent quadrants (two of four possible combinations represented) [25]

6.5.2 Assignment over Obstacles

However, if the optimal Steiner location overlaps an obstacle the simple assignment is impossible, as depicted in Fig. 6.14. The solution found is to generate, the maximum of six possible new Steiner locations: four correspond to points assigned in the same manufacturing layer, i.e., on left, right, above and below of the obstacle obstructing the Steiner assignment; and two correspond to the points in the exact (x, y) coordinates of the optimal Steiner point, but located one layer above (x, y, z + 1) or one layer below (x, y, z − 1), if possible, as presented in Fig. 6.14c. The paths through all the possible Steiner points are computed and the shortest path, i.e., three edges and the Steiner point, is selected. If two paths have the same wire length, the preferred path is the one that maximizes the overlapped length between I_i and I_j.

The output of the global router, after multiport selection and Steiner point assignment is illustrated in Fig. 6.15, where the port-to-port optimal rectilinear forest for the MST shown in Fig. 6.2b, considering four rectangular obstacles shown in gray, is depicted. Two Steiner points were assigned to minimize the total wire area under the EM.

6.5.3 Symmetry Considerations

The previous proceeding provides satisfying results for general wiring (i.e., wires without symmetries), but for pairs of symmetric wires especial considerations must be taken. The fundamental principle is that a wire of a pair of symmetric wires cannot be 'Steinered' with a wire that does not belong to a SP. Two distinct situations can happen:

- A Steiner is introduced for each symmetric pair and the current-flow is incremented recursively to the previous overlapped sections as more edges are added. All the previous symmetries are respected in this process and the resulting wire

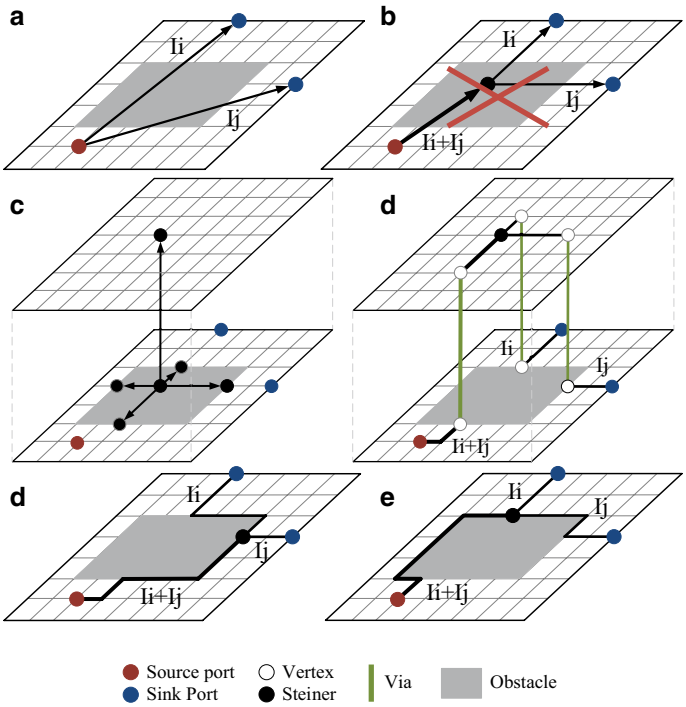


Fig. 6.14 Steiner assignment with obstacles: (**a**) Two edges with common source and different sinks, (**b**) The optimal Steiner location is impossible. (**c**) New Steiner points, 5 available. (**d**) Path length: $12+3*V_{cost}$, overlapped: $4+1*V_{cost}$, (**e**) Path length: 15, overlapped: 7, (**f**) Path length: 15, overlapped: 7 [25]

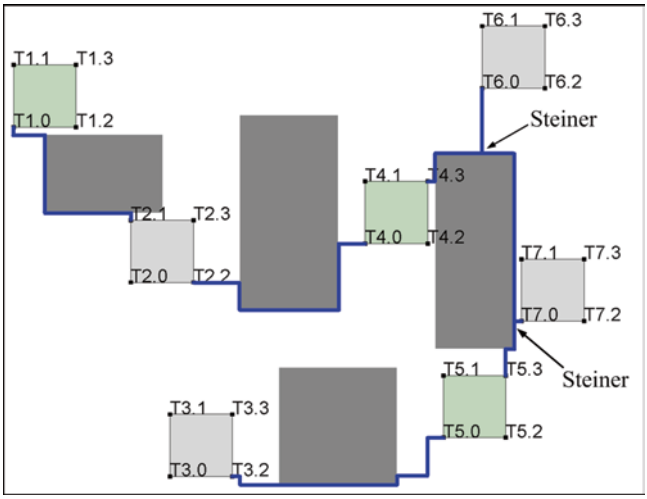


Fig. 6.15 Multiport selection and Steiner points assigned for the multiport topology of Fig. 6.2b, considering four obstacles [25]

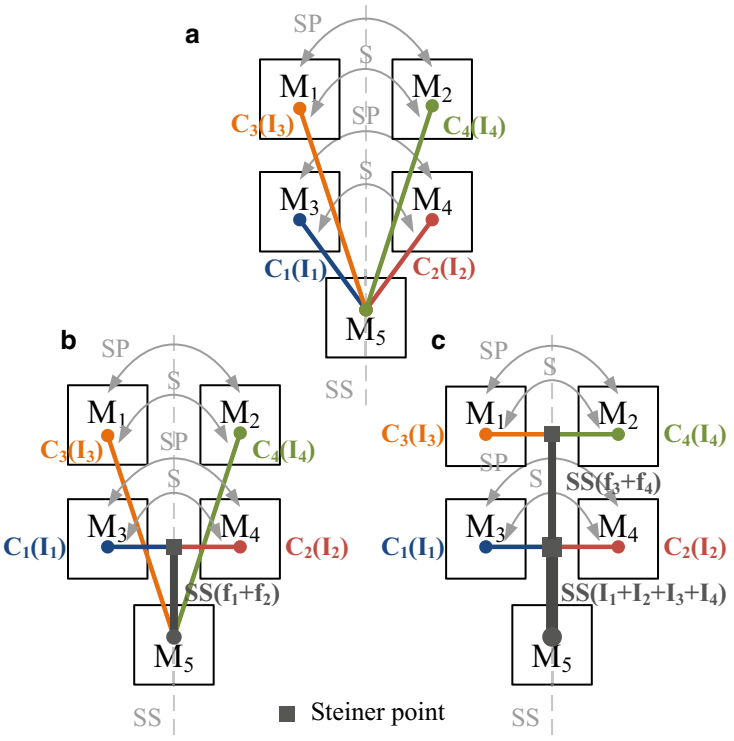


Fig. 6.16 (**a**) Four edges with common source and different sinks, S{C_1, C_2} and S{C_3, C_4} are the symmetric wire pairs. (**b**) Steiner introduced. A self-symmetric wire is created from the common source to the Steiner point. (**c**) A second Steiner point is introduced. Steiner represented by the *black square*

connecting the Steiner point is set as a self-symmetric wire. This situation is common in current intensive nets like *vdd* or *vss* due to the nature of the EM-aware WT constructed. Since the flow of electric-current-sources is only redirected to the electric-current-sinks (no source-to-source or sink-to-sink flow is allowed), a single terminal (*vdd* or *vss* pin) usually connect all the remaining terminals of the net. In Fig. 6.16 are represented four edges with the same source and different sinks, the symmetric wire pairs are: S{C_1, C_2} and S{C_3, C_4}.

- The second situation can be depicted in Fig. 6.17. Unlike the previous, in this heuristic four directed edges must be considered at a time, and the difference is that two wires from different symmetric wire pairs are 'Steinered'. However, all the previous symmetries are respected in this process and new symmetric wire pairs are created for the two connections to the new Steiner points.

Note that different current-densities for the two wires of a symmetric pair, or, for multiple overlapped areas, are supported.

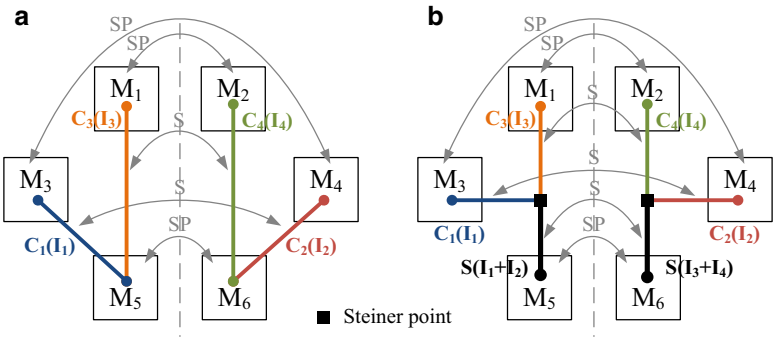


Fig. 6.17 (**a**) Four edges, two common sources and four different sinks, S{C_1, C_2} and S{C_3, C_4} are the symmetric wire pairs. (**b**) Two Steiner points introduced. Steiner represented by the *black square*

6.6 Detailed Router

The last step of the proposed methodology is the detailed Router, which is responsible for computing the wires' width according to the electric-currents imposed on them (Sect. 6.2.1) and ensures design rule- and layout-versus-schematic-correctness.

Expanding all the obstacles boundaries in the multilayer grid with the minimum space requirement and half width of the segments to ensure design rule compliant layouts is not viable in an EM-aware methodology, due to the multitude of different wires' widths considered. Setting the obstacles' boundaries for the maximum wire's width found in the EM-aware WT will result in a harsh overdesign and consequently the optimal minimum area routing could not be achieved, or, maybe even a valid solution. On the other hand, setting that value with the minimum width will certainly result in solutions containing design rule errors and/or short-circuits among the different nets.

The solution found, was to allow the rectilinear paths, which until this phase were kept in the grid introduced in Sect. 6.4.2, to go off-grid in the detailed routing phase, obviously over the manufacturing grid of the technology process. Terminal-to-terminal connections, rectilinear paths, Steiner points and symmetry information previously attained are used as starting point for an optimization process. Previously routed nets are not used as blockage which avoids the construction of computationally expensive detailed grids every time a new wire needs to be routed. All wires of all nets are optimized simultaneously, and this way the quality of the routing solutions does not depend of the traditional heuristics for net (re)ordering, backtracking or re-routing. The wires' widths are computed on-the-fly, as the effective width is a function of technology inherent parameters that change for the actual level of conductor (metal) assigned for the wire and the wire's length. This process maintains the wiring symmetry during all the optimization process, leading to a correct and symmetric final layout.

Table 6.1 Summary of constraints and objectives

	Description
Constraint	
$g_1(x)$ = SCC	Number of short circuits
$g_2(x)$ = DRC	Number of violations of the design rules (minimum distances, etc.)
$g_3(x)$ = ERC_1	Crossings between noisy and sensitive nets
$g_4(x)$ = ERC_2	Number of noisy/sensitive nets running on top of the devices' active areas
Objective	
$f_1(x)$ = wiring_ length	Total wiring length computed with the associated conductors cost
$f_2(x)$ = −crosstalk	Total distance between noisy and sensitive nets

6.6.1 Evolutionary Algorithm

The optimization kernel used is a modified implementation of the multi-objective multi-constraint evolutionary algorithm NSGA-II [21]. Unlike the traditional approaches, not all the design rules are enforced in a correct-by-construction manner. Instead, the Router follows an optimization-based approach where the algorithm performs slight structural and layer changes in a population of different and independent routing solutions. The different routing solutions are evaluated by a built-in layout evaluation algorithm that is basically a lightweight design rule check (DRC), layout versus schematic (LVS) and electrical rule check (ERC) procedures, which provides at each generation the constraints for the optimization problem. The solutions should converge to feasible, i.e., feasibility is attained when no design rule, short-circuit or any other electrical-rule is violated, while keeping the minimum wire length, and/or other design objectives. This is performed by keeping the better solutions in the population for further generations and the constraints gradually driven to zero. The multi-objective multi-constrained optimization kernel module was designed to solve the problem:

$$\begin{aligned} &find\, x\, that \min imizes\ f_m(x) \quad m = 1,2,\ldots,M \\ &subject\, to\, g_j(x) = 0 \quad j = 1,2,\ldots,J \end{aligned} \tag{6.17}$$

where, x is a vector of N optimization variables, $g_j(x)$ one of the j constraints and $f_m(x)$ one of the M objective functions.

A summary of the constraints and possible objectives used by the multi-objective evolutionary algorithm are depicted in Table 6.1, other constraints or objectives can be easily added as suited for the designer. Due to the difficulties found on automatically defining the sensitivity of the signal nets, they can be marked manually as noisy or sensitive, and will receive special treatment during the detailed routing procedure.

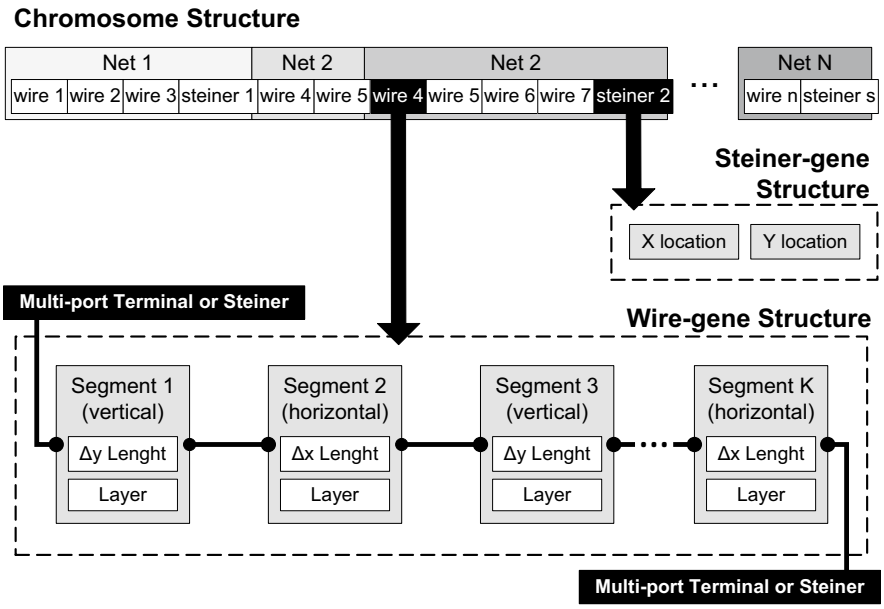


Fig. 6.18 Chromosome structure with *n* wires and *s* Steiner points distributed by N different nets; Wire-gene structure composed of K-even linked segments; and Steiner-gene structure

6.6.2 Chromosome Structure

Each element in the population, a chromosome, encodes the information about a complete and different routing solution. Two different genes types are available, henceforward called wire-genes and Steiner-genes, as depicted in Fig. 6.18.

A wire-gene corresponds to a different terminal-to-terminal instance attained in the global Router. A wire-gene structure with K-even segments (K/2 horizontal and K/2 vertical segments) can be inferred from Fig. 6.18. Every wire-gene in a chromosome has the same number of K-even segments linked segments, no matter the complexity of the wire mapped to it, which eases the implementation of the genetic operators. Each wire has the information of the source and sink terminal, which consists on the exact location of each of the available terminal's ports or Steiner. By its part, each segment has an associated length and layer, and the segments with even indexes are vertical and the ones with odd indexes are horizontal. Zero length segments are legal, e.g., a wire that starts with a horizontal segment has the length of segment 1 set to zero. A Steiner-gene can be seen as regular terminal, but with only one available port. Based on this complex chromosome structure the optimization of all wires of all nets simultaneously is possible, while exploring the multiport terminals.

The number of optimization variables which can be computed by Eq. (6.18), lead to a huge dimension of the search space:

$$x^d = \sum_{i=1}^{N} \left[\sum_{j=1}^{n_i} \left(\sum_{k=1}^{K} \left(lenght_{ijk} + layer_{ijk} \right) + SourcePort + SinkPort \right) + \sum_{j=1}^{s_i} \left(x_j + y_j \right) \right] \tag{6.18}$$

where, N is the number of nets, n_i the number of wires in the net i, and K stands for the fixed number of segments in wire j. $length_{ijk}$/$layer_{ijk}$ is the length/layer of the segment k of wire j of net i. Finally, s_i is the number of Steiners points in the net i.

Each detailed routing phase is preceded by an initialization step, which consists on generation of the initial population of chromosomes. Two different methods for wire initialization are used: global-Router-based and greedy initialization. In the global-Router-based initialization, all wire-genes of a chromosome are initialized with the paths obtained in the global routing phase. This step is essential to provide the theoretical optimal starting point for the optimization. For the greedy initialization, only the connectivity is used, and the two closest ports of the terminals connecting are selected as starting point. Additional detail on the initialization on the chromosome and the genetic operators applied to it is remitted to the *Router* Chapter of [22] and [26].

6.6.3 Optimization Phases

As overviewed in the previous Sections, the planning phases (Sects. 6.2 and 6.3) and the Global routing phases (Sects. 6.4 and 6.5) are executed independently for each different net contained in the netlist. The exceptions are the wiring symmetry considerations that can be taken between wires of different nets. The detailed Router is actually a set of different instances of the optimization kernel, as depicted in Fig. 6.19, first, single-net detailed optimization procedures, and only after, the multi-net detailed optimization.

The single-net procedure is executed N times independently, where N is the number of nets to be routed. This single-net optimization eliminates any design rule, short circuit or electrical rule violation present on the previous steps, and provides the theoretical optimal solution for the routing of that net (when ignoring the presence of other conflicting nets in the layout). For the current architecture, the existence of different optimization stages is due to the fact that when the exact physical representation of the wires is generated the number of layout elements being handled greatly increases and the number of constraints violated grows quickly. By dealing with each net independently the quality of the final solution is greatly increased.

To generate the initial population of the multi-net procedure the optimization results of the single-net procedures are used. The resultant pre-optimized nets are all combined. The use of pre-optimized nets provides a good starting point for the MP/MT detailed Router. However, when combining all the optimized nets, the solution

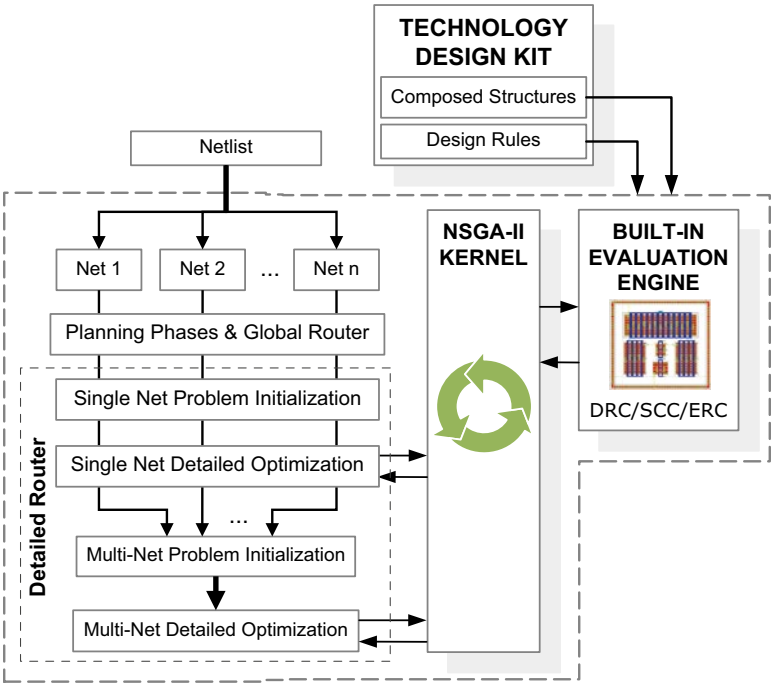


Fig. 6.19 Different optimization phases used during detailed Router procedure

is not necessarily compliant with the design rules, at this point the evolutionary optimization kernel is used again, but now globally, to drive to zero the number of design rule violations and minimize the total-interconnect length, restricted, if desired, by an optional set of sensitivity constraints.

The evaluation of the population is done using a built-in validation engine, which uses a thread-pool for parallel processing and provides all the constraints that guide the optimization, properly detailed in next Section. Also, the module generator is used during this detailed routing process to provide the design rules and the required composed structures (e.g., contact/vias stacks, power nets, etc.). Additional detail on the built-in evaluation procedure is remitted to the *Router* Chapter of [22].

6.7 Conclusion

In the fully-automatic Router, AIDA-L replaces the designer's obligation to describe the exact routing or providing the high level floorplan guidelines, by implementing a set of comprehensive wire planning and routing steps. This extremely versatile approach inputs only the floorplan, the netlist and a set of electric-currents for each terminal, and the tool automatically starts the procedure that will lead to a solution

that strictly complies with the target technology design rules. It is allow fundamental to perform layout-aware optimizations in the AIDA's framework [23, 24].

Dealing with a multitude of different electric-currents manually and WS is a time-consuming and error-prone task. It is not only hard to manually sketch a symmetric current-correct WT, but even slight changes in the floorplan disposition or in the design specifications may discard all the previous wire planning and/or routing. Also, non-idealities of the layout are becoming increasingly more relevant and must be taken into account in the design of the interconnects of AMS ICs.

AIDA-L's Router features a MP/MT EM-aware and IR-drop-avoidance routing methodology, also considering WS for the generation of analog IC layouts. It intends to bypass the limitations of the available approaches by automatically designing symmetric, electric-current-correct and design rule-correct nets, and the use of MP allows entwining the stages of the automatic wiring planners/routers with realistic device geometries. The implemented design flow solves the wire planning, MP selection, path rectilinearization, Steiner point assignment and detailed routing automatically, to minimize the total wiring area under several constraints.

References

1. XML, http://www.xml.com/
2. R. Martins, N. Lourenço, N. Horta, LAYGEN II: Automatic analog ICs layout generator based on a template approach, in *Genetic and Evolutionary Computation Conference (GECCO)*, Philadelphia, PA, July 2012, pp. 1127–1134
3. R. Martins, N. Lourenço, N. Horta, Multi-objective multi-constraint routing of analog ICs using a modified NSGA-II approach, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sept 2012, pp. 19–21
4. J.R. Black, Electromigration: A brief survey and some recent results. IEEE Trans. Electron Devices **16**(4), 338–347 (1969)
5. J. Lienig, Electromigration-aware physical design of integrated circuits, in *18th International Conference on VLSI Design*, Jan 2005, pp. 77–82
6. J. Lienig, Introduction to electromigration-aware physical design, in *Proc. International Symposium on Physical Design (ISPD)*, Mar 2006, pp. 39–46
7. 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
8. J. Lienig, G. Jerke, T. Adler, Electromigration avoidance in analog circuits: Two methodologies for electromigration-driven routing, in *International Conference on VLSI Design Conference*, Jan 2002, pp. 372–378
9. J. Lienig, G. Jerke, Current-driven wire planning for electromigration avoidance in analog circuits, in *Proceedings of the Asia-South Pacific Design Automation Conference (ASP-DAC)*, Jan 2003, pp. 783–788
10. 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
11. I. Jiang, H.-Y. Chang, C.-L. Chang, WiT: Optimal wiring topology for electromigration avoidance. IEEE Trans. Very Large Scale Integr. Syst. **20**(4), 581–592 (2012)

12. B. Cherkassky, A. Goldberg, Negative-cycle detection algorithms. Math. Program. **85**(2), 277–311 (1999)
13. I. Jiang, H.-Y. Chang, C.-L. Chang, Optimal wiring topology for electromigration avoidance considering multiple layers and obstacles, in *Proc. International Symposium on Physical Design (ISPD)*, Mar 2010, pp. 177–184
14. J. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Am. Math. Soc. **7**(1), 48–50 (1956)
15. Y. Yang, I. Jiang, Analog placement and global routing considering wiring symmetry, in *11th International Symposium on Quality Electronic Design (ISQED)*, Mar 2010, pp. 618–623
16. M. Eick, M. Strasser, K. Lu, U. Schlichtmann, H. Graeb, Comprehensive generation of hierarchical placement rules for analog integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **30**(2), 180–193 (2011)
17. H.E. Graeb (ed.), *Analog layout synthesis: A survey of topological approaches* (Springer, Berlin, 2010)
18. G. Robins, A. Zelikovsky, Minimum steiner tree construction, in *The Handbook of Algorithms for VLSI Physical Design Automation*, ed. by C. Alpert, D. Mehta, S. Sapatnekar (CRC Press, Taylor & Francis Group, New York, 2009)
19. C. Helvig, G. Robins, A. Zelikovsky, New approximation algorithms for routing with multiport terminals. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **19**(10), 1118–1128 (2000)
20. P. Hart, N. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
21. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
22. R. Martins, N. Lourenço, N. Horta, *Generating Analog IC Layouts with LAYGEN II*. SpringerBriefs in Applied Sciences and Technology (Springer, New York, 2013)
23. N. Lourenço, R. Martins, N. Horta, Layout-aware synthesis of analog ICs using floorplan & routing estimates for parasitic extraction, in *Design, Automation & Test in Europe Conference (DATE)*, Mar 2015, pp. 1156–1161
24. R. Martins, N. Lourenço, A. Canelas, R. Póvoa, N. Horta, AIDA: Robust layout-aware synthesis of analog ICs including sizing and layout, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sept 2015, pp. 1–4
25. R. Martins, N. Lourenço, A. Canelas, N. Horta, Electromigration-aware analog Router with multilayer multiport terminal structures. Integr. VLSI J. **47**(4), 532–547 (2014). Reprinted from Integration, the VLSI Journal. With permission from Elsevier
26. R. Martins, N. Lourenço, N. Horta, Routing analog ICs using a multi-objective multi-constraint evolutionary approach. Analog. Integr. Circ. Sig. Process **78**(1), 123–135 (2014). doi:10.1007/s10470-013

Chapter 7
Empirical-Based Parasitic Extractor

The developed Parasitic Extractor targets layout-aware sizing methodologies due to its ability to run accurately not only over the complete layout, but also over the incomplete layout, which has only the non-detailed routing (i.e., the global routing), greatly reducing the required computational time. The advantages of using this simplified approach for layout in the loop are clear in terms of computational efficiency. However, designers' must have the accuracy of industry "standard" extraction tools, hence, it is mandatory to have high accuracy in the values of the parasitic structures identified, even in a simplified extraction procedure. On the other side, it should be practical to use, hence a minimum set of inputs is considered. The empirical-based Parasitic Extractor inputs only the floorplan solution generated by the Placer, the global routing solution, and the *intercap* models provided by the foundry that contain the standard interconnect capacitance values.

The first section of this chapter covers the general description of the empirical-based Parasitic Extractor architecture. The processing of the *intercap* models tables is addressed in section "*Intercap* Models Processing", while in section "RC Extraction" the extraction of the resistive and capacitive structures is detailed.

7.1 Empirical-Based Parasitic Extractor Architecture

From the parasitic extractors integrated in layout-aware methodologies, overviewed in Chap. 2 of this book, it is possible to found three different approaches: 1/2-D models [1], analytical polynomial models [2–4] and external extractors [5–9]. The proposed 2.5-D methodology, whose architecture/design flow is shown in Fig. 7.1, depicts the main tasks performed by the Parasitic Extractor.

In the Parasitic Extractor, prior knowledge of the circuit's parasitics is not required in order to estimate all resistance and capacitance parasitic structures. To achieve this, first, in the *Intercap* Models Processing (Sect. 7.2), the tables with the capacitive values provided by the foundry are processed using a simple but effective

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_7

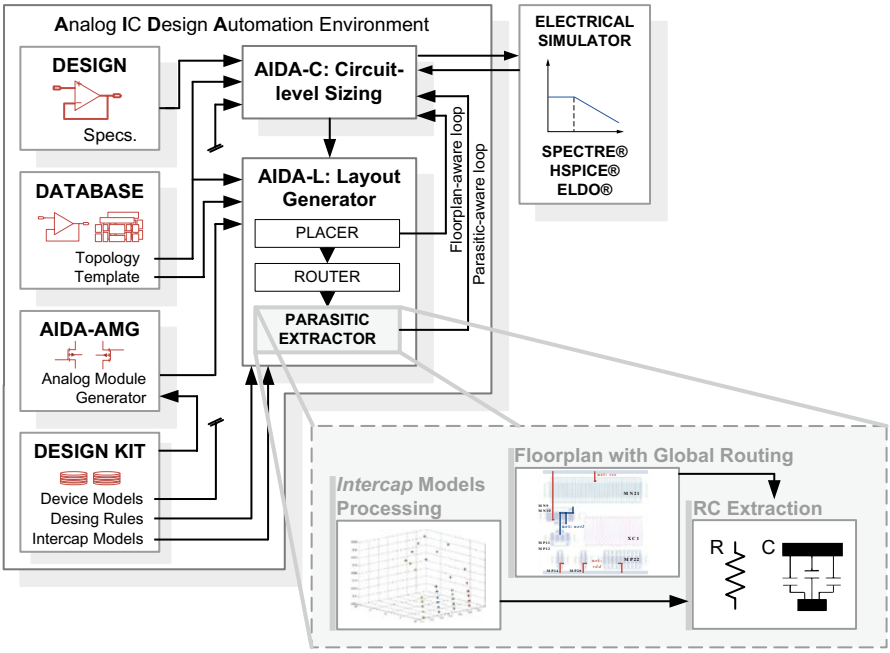


Fig. 7.1 Empirical-based Parasitic Extractor: *Intercap* Models Processing and RC Extraction blocks embedded in the AIDA-L's design flow

linear-by-segments interpolation. Then, in the RC Extraction block (Sect. 7.3), the resistive and capacitive parasitics are estimated from the incomplete layout (i.e., floorplan and global routing) using straightforward empirical-based models combined with 2.5-D geometric considerations.

7.2 *Intercap* Models Processing

As stated, the developed Parasitic Extractor follows an empirical-based approach where the parasitic capacitances are computed using the data provided by the foundry. These *intercap* models are tables with precise empirical data measured experimentally and provided one for each pair of layers, for the three primitive capacitance classifications: area capacitance (C_a), fringe capacitance (C_f) and coupling capacitance (C_c), as schematized in Fig. 7.2. Figure 7.3 presents the capacitance coefficients for the *metal1* layer above *substrate*, which are function of the reference conductor's *width*, and also, the *space* value to the closest conductor on the same layer, for the United Microelectronics Corporation (UMC) 130 nm design process.

The *intercap* tables provide several reference points for *width* and *space* values, namely, four different values of *width*, and eight different values of *space* for each

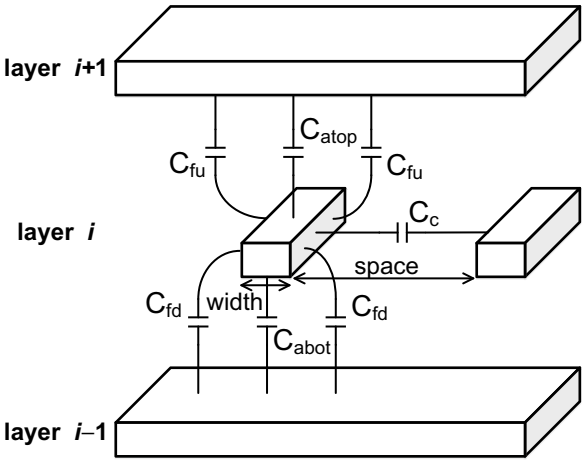


Fig. 7.2 The different primitive capacitances: C_{atop}, area capacitance to top ground plane; C_{abot}, area capacitance to bottom ground plane; C_{fu}, fringe up capacitance coefficient; C_{fd}, fringe down capacitance coefficient; and C_c, coupling capacitance to adjacent conductor. Capacitive components are function of the conductor's width and space to the closest conductor on the same layer

[metal1,above,substrate](typical)				
width(um)	space(um)	Ca(fF/um)	Cf(fF/um)	Cc(fF/um)
0.1600	0.1600	9.□□□□E-03	5.□□□□E-03	1.□□□□E-01
0.1600	0.3200	9.□□□□E-03	9.□□□□E-03	7.□□□□E-02
0.1600	0.4800	8.□□□□E-03	1.□□□□E-02	4.□□□□E-02
0.1600	0.6400	8.□□□□E-03	1.□□□□E-02	3.□□□□E-02
0.1600	0.8000	8.□□□□E-03	1.□□□□E-02	3.□□□□E-02
0.1600	3.0400	8.□□□□E-03	3.□□□□E-02	5.□□□□E-03
0.1600	4.6400	8.□□□□E-03	3.□□□□E-02	2.□□□□E-03
0.1600	6.2400	8.□□□□E-03	3.□□□□E-02	1.□□□□E-03
0.3200	0.1600	1.□□□□E-02	7.□□□□E-03	1.□□□□E-01
0.3200	0.3200	1.□□□□E-02	1.□□□□E-02	6.□□□□E-02
0.3200	0.4800	1.□□□□E-02	1.□□□□E-02	4.□□□□E-02
0.3200	0.6400	1.□□□□E-02	1.□□□□E-02	3.□□□□E-02
0.3200	0.8800	1.□□□□E-02	2.□□□□E-02	2.□□□□E-02
0.3200	1.1200	1.□□□□E-02	2.□□□□E-02	2.□□□□E-02
0.3200	4.4800	1.□□□□E-02	3.□□□□E-02	3.□□□□E-03
0.3200	9.2800	1.□□□□E-02	4.□□□□E-02	1.□□□□E-03
0.4800	0.1600	2.□□□□E-02	7.□□□□E-03	1.□□□□E-01
0.4800	0.3200	2.□□□□E-02	1.□□□□E-02	7.□□□□E-02
0.4800	0.4800	2.□□□□E-02	1.□□□□E-02	5.□□□□E-02
0.4800	0.6400	2.□□□□E-02	1.□□□□E-02	4.□□□□E-02
0.4800	0.8000	2.□□□□E-02	1.□□□□E-02	3.□□□□E-02
0.4800	1.4400	2.□□□□E-02	2.□□□□E-02	1.□□□□E-02
0.4800	5.9200	2.□□□□E-02	4.□□□□E-02	2.□□□□E-03
0.4800	12.3200	2.□□□□E-02	4.□□□□E-02	7.□□□□E-04
0.6400	0.1600	2.□□□□E-02	7.□□□□E-03	1.□□□□E-01
0.6400	0.3200	2.□□□□E-02	1.□□□□E-02	7.□□□□E-02
0.6400	0.4800	2.□□□□E-02	1.□□□□E-02	5.□□□□E-02
0.6400	0.6400	2.□□□□E-02	1.□□□□E-02	4.□□□□E-02
0.6400	0.9600	2.□□□□E-02	2.□□□□E-02	2.□□□□E-02
0.6400	1.7600	2.□□□□E-02	3.□□□□E-02	1.□□□□E-02
0.6400	7.3600	2.□□□□E-02	4.□□□□E-02	1.□□□□E-03
0.6400	15.3600	2.□□□□E-02	4.□□□□E-02	5.□□□□E-04

Fig. 7.3 Example of an *intercap* model for *metal1* above *substrate*, for the UMC 130 nm design process. Values omitted from the table are property of the United Microelectronics Corporation

one of those widths. However, it is still required to process this data to obtain the capacitance values for the intermediate points. The C_a, C_f and C_c values for *intercap* table of Fig. 7.3 for the *metal1* above *substrate* conditions are graphically represented in function of *width* and *space* in Fig. 7.4.

By observing the plots, it is clear that the area capacitance is strongly dependent of the variation of *width* values, while almost constant with respect to the *space* values. This major dependence in only one of the variables is also verified for fringe and coupling capacitances, however, in these cases the dependency is in terms of *space* values and the impact of *width* variation is almost negligible. In order to simplify the sampling of the *intercap* tables, the variable with less impact is discarded from the model and the data is interpolated linearly-by-segments.

In Fig. 7.5 the linear-by-segments interpolations of C_a, C_f and C_c in function of only one variable are presented. The line corresponds to the approximations made, and all the points of Fig. 7.5 are plotted in the corresponding 2-D projection. The proximity of the original points from the linear-by-segments interpolation justify the approximations made, without adding relevant error to the estimation, and avoids complicated nonlinear interpolations/regressions. Although a straightforward interpolation is performed on the foundry's *intercap* models, a more sophisticated method of regression into a polynomial or posynomial model could guarantee a stronger portability through a different behavior of the *intercap* models when migrating to another technology. However, this feature is left for future revisions of the Parasitic Extractor as all the remaining *intercap* models for the UMC 130 nm present similar behavior.

7.3 RC Extraction

In this sub-section, the estimated parasitic resistances of the interconnects, parasitic bulk capacitances, parasitic capacitances between terminals' shapes of the devices, between terminals' shapes and interconnects, and between pairs of global routing interconnects are detailed. To avoid an external extractor, reliable estimates for the parasitic devices must be derived internally. For MOS transistors it is common the use of geometric methods, where the device's width, length and number of fingers are used in an equation that provides the estimated parasitic. However, the technology-dependency of those equations, the difficulty found to derive them for more complex layout styles, e.g., common centroid, and the inapplicability to wires, pressed for a different approach. Hence the same procedure used for wires is applied for the intra-device parasitics.

The procedure follows the loops outlined in Algorithm 7.1, the extraction of each parasitic capacitance is made with the geometrical considerations and equations introduced in the following sections.

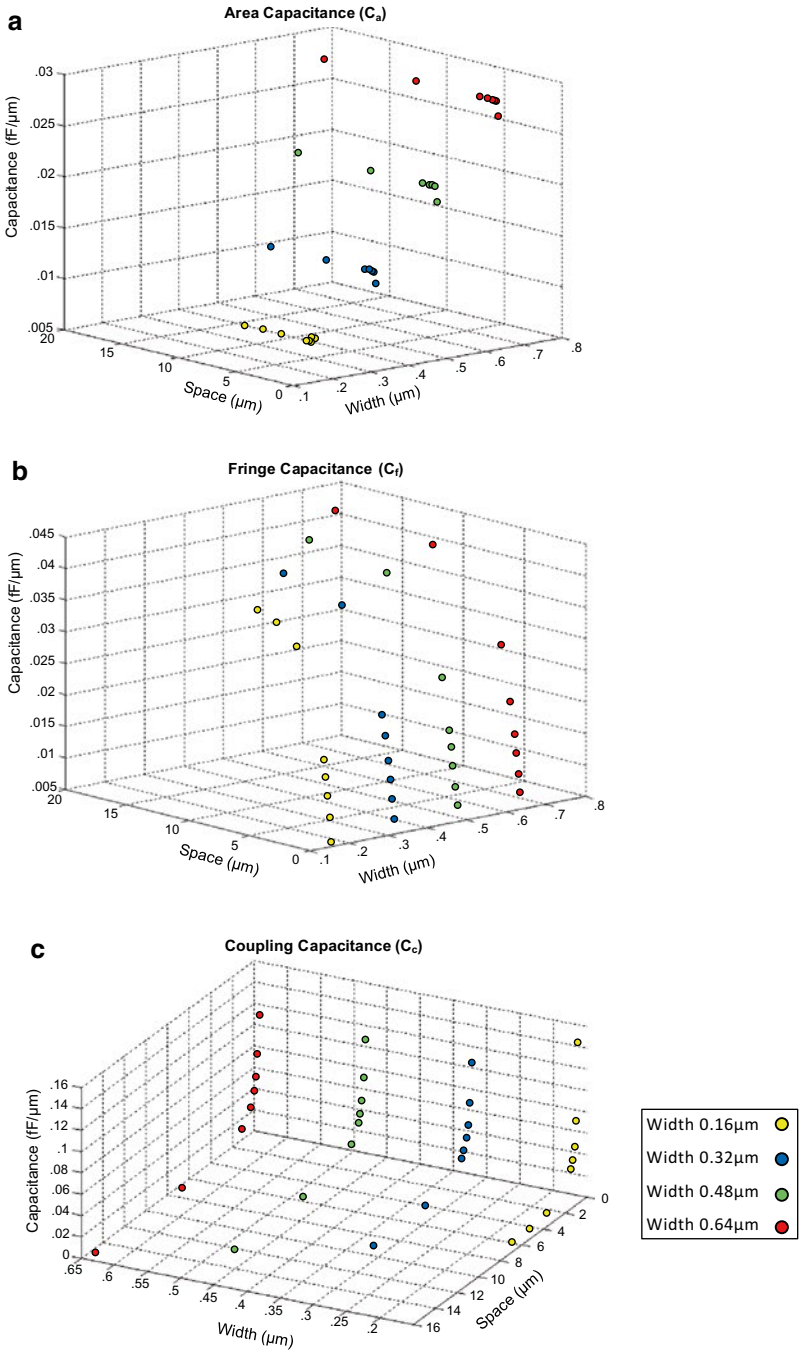


Fig. 7.4 3-D Representation of the *metal1* above *substrate intercap* model for the UMC 130 nm design process: (**a**) area capacitance C_a, (**b**) fringe capacitance C_f and (**c**) coupling capacitance C_c

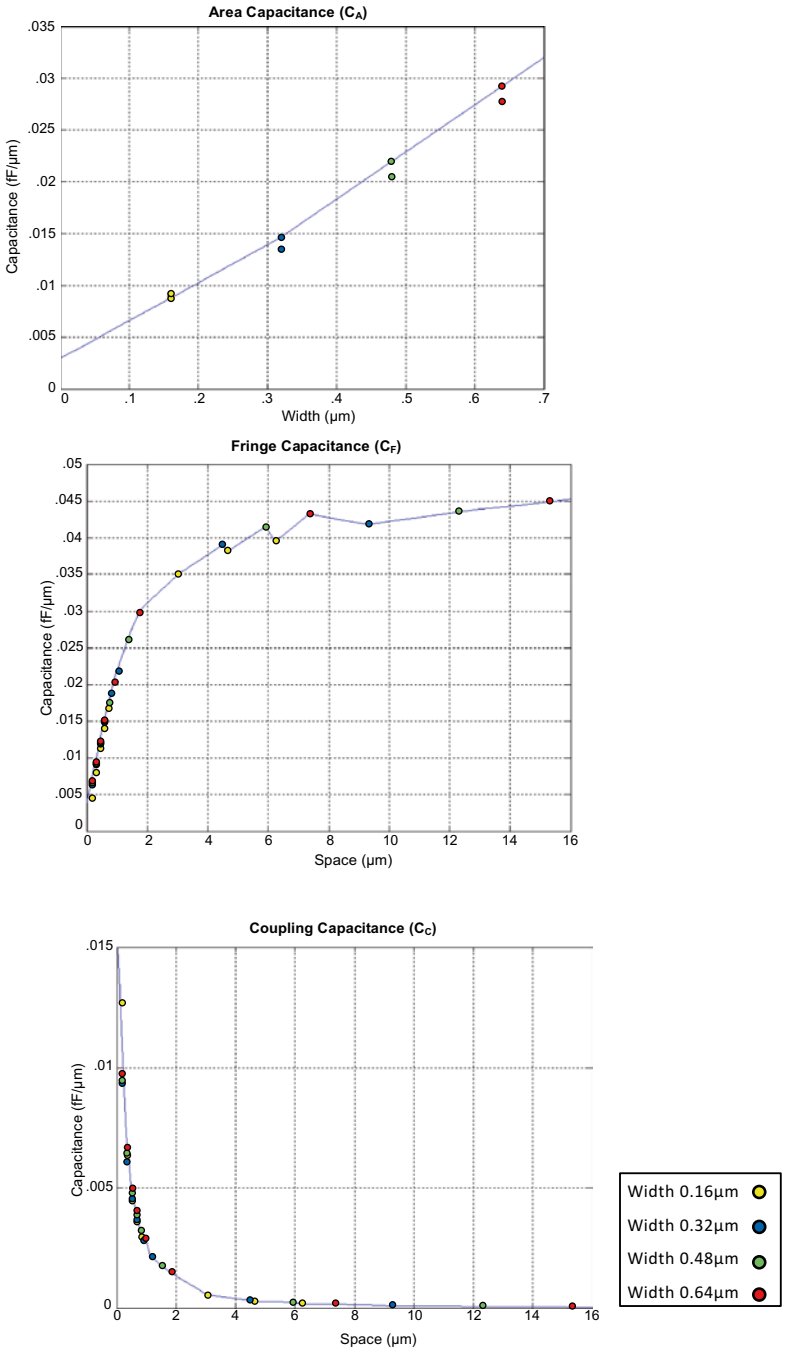


Fig. 7.5 Linear-by-segments interpolation of the *metal1* above substrate *intercap* model for the UMC 130 nm design process: (**a**) area capacitance C_a in function of width, (**b**) fringe capacitance C_f and (**c**) coupling capacitance C_c in function of space

Algorithm 7.1: RC Extraction

input:	**List<Devices<**Terminal<Shapes**>>>** *Floorplan* (shapes in the floorplan) **List<Net<List<Edge<**$w_{i,j}$**,** **List<**Nodes**>>>>>** *GlobalRouting* (global routing solution, each Edge represents a terminal-to-terminal connection defined by a path **List<**Nodes**>**)
1.	//Parasitic Resistance
2.	**for** each *net i* in the *GlobalRouting* **do**
3.	**for** each *Edge j* in the net *i* **do**
4.	**for** Node *k* to Node *k+1* of *Edge j* **do**
5.	*pres*:= compute parasitic of the segment resistance by square counting, with wire's width $w_{i,j}$ and wire's length as a distance between Node *k* to Node
6.	*k+1* increment *k* and add *pres* to *pres_total*
7.	**return** *pres_total* of *Edge j*
8.	//Parasitic Capacitances between Terminals
9.	**for** each *Device i* in the *Floorplan* **do**
10.	**for** each *Terminal m* in the *Device i* **do**
11.	**for** each *Device j* in the *Floorplan* **do**
12.	**for** each *Terminal n* in the *Device i* **do**
13.	with *m* != *n*
14.	**for** each *Shape p* in the *Terminal m* **do**
15.	**for** each *Shape q* in the *Terminal n* **do**
16.	*pcap*:= compute parasitic capacitance between *p* and *q*
17.	*pcap_total:*= increment *pcap* value
18.	**return** *pcap_total* between *Terminal m* and *Terminal n*
19.	//Parasitic Capacitances between Edges
20.	**for** each *net i* in the *GlobalRouting* **do**
21.	**for** each *Edge m* in the net *i* **do**
22.	**for** each *net j* in the *GlobalRouting* **do**
23.	**for** each *Edge n* in the net *j* **do**
24.	with *i* != *j* and *m* != *n*
25.	**for** Node *p* to Node *p+1* of *Edge m* **do**
26.	**for** Node *q* to Node *q+1* of *Edge n* **do**
27.	*pcap*:= compute parasitic capacitance between
28.	segment (*p*, *p+1*) and segment (*q*, *q+1*)
29.	*pcap_total:*= increment *pcap* value
30.	increment *q*
31.	increment *p*
32.	**return** *pcap_total* between *Edge m* and *Edge n*
33.	//Parasitic Capacitances between Terminals and Edges
34.	//Parasitc Capacitances to the Substrate
35.	//follow the same principles above for terminals or edges processing

7.3.1 Parasitic Resistance

The parasitic resistance $R_{parasitic}$ for each wire's segment, which is defined by a *width* and *length*, is computed from the resistance per unit square R_c tabulated for a given conductor at a temperature T:

$$R_{parasitic} = R_C(T) \times \frac{length}{width} \tag{7.1}$$

Each wire has any number of segments/bends, so all partial resistance components are considered for square counting, as illustrated on Fig. 7.6.

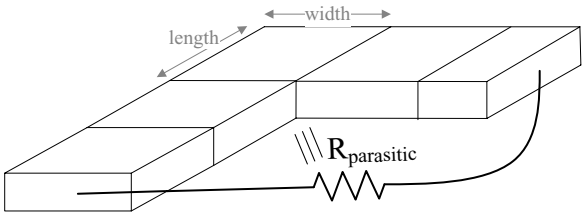


Fig. 7.6 Parasitic interconnect resistance computed by square counting

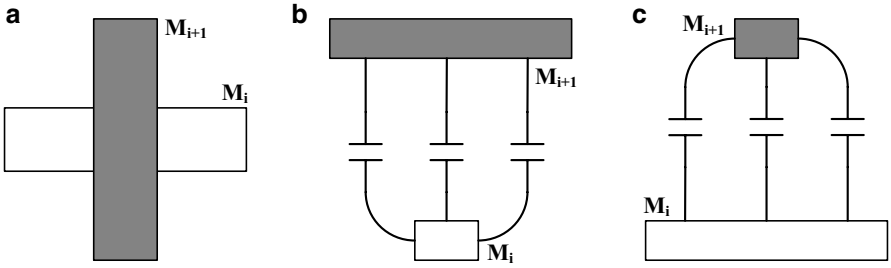


Fig. 7.7 3-D Modeling as a combination of two 2-D structures: (**a**) top view, (**b**) cross section view 1, and (**c**) cross section view 2 [10]

7.3.2 Parasitic Capacitances

For capacitive parasitic extraction the alternatives found in the literature are: the computationally expensive numerical methods, which are hardly usable within the optimization loop; approximated analytic methods obtained from electromagnetic models; or empirical-based techniques, where the extensive experimental data provided by the foundry for a particular technology is used and fitted to a certain model [13]. In this work, an empirical-based technique where the technology-dependent parameters obtained by consulting the linear-by-segments interpolations derived in the previous section is used.

From the methodologies found in the literature for capacitive extraction, in 1-D extraction models only the C_a component is weighted into an expression that is function of interconnects' area and perimeter. Then, 2-D extraction models extend 1-D to also account for capacitive effects in the same layer, i.e., C_c component. Still, the effect of 3-D structures, e.g., two wires crossing in different conductors, are not covered in 2-D approaches. However, due to the countless geometrical considerations and the computational effort that C_f component requires, complete 3-D capacitance extractors are not trivial extensions of the 2-D. An extension to address 3-D effects is the 2.5-D method, where several realistic 3-D effect are modeled as a combination of two orthogonal 2-D structures [10], as it is illustrated in Fig. 7.7. In the proposed Parasitic Extractor, by carefully composing a 3-D solution from the two orthogonal 2-D ones, most 3-D effects are captured, while avoiding the complicated 3-D evaluation in the simplified procedure.

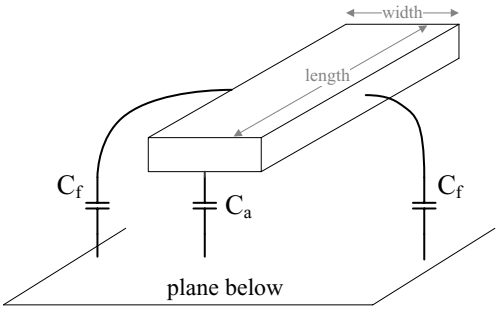


Fig. 7.8 Parasitic capacitance to the plane below $C_{substrate}$: C_a and C_f components

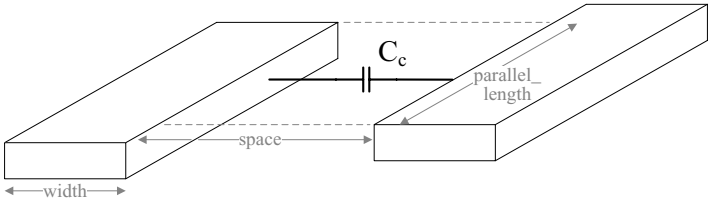


Fig. 7.9 Parallel capacitance between two conductors on the same layer $C_{lateral}$: C_c component

7.3.2.1 Substrate Capacitance

Each device's terminal (or interconnect) has an intrinsic capacitance $C_{substrate}$ considered to the plane below, whether it is the substrate, well or active area. To compute the intrinsic/substrate capacitance, two types of capacitances must be considered: C_a, which covers the capacitances on the overlap area, right below the conductor, and C_f which considers the remaining field lines, as presented in Fig. 7.8. $C_{substrate}$ is computed by the sum of the area and the two fringe components according to:

$$C_{Substrate} = \left(C_a\left(width\right) \times length\right) + \left(2 \times Cf\left(space\right) \times length\right) \tag{7.2}$$

where the C_a component is obtained by interpolation of the technology-dependent data provided in the foundry's *intercap* models as a function of the conductor's *width*. The fringe component C_f is obtained by interpolation with the *space* between the reference shape and the closest shape in the same layer.

7.3.2.2 Lateral Capacitance

The lateral capacitance $C_{lateral}$ is considered between two conductors running in parallel or perpendicular on the same layer, as presented in Fig. 7.9. Here, the fringe components are negligible, and $C_{lateral}$ is defined as:

$$C_{lateral} = C_c\left(c\_space\right) \times parallel\_length \tag{7.3}$$

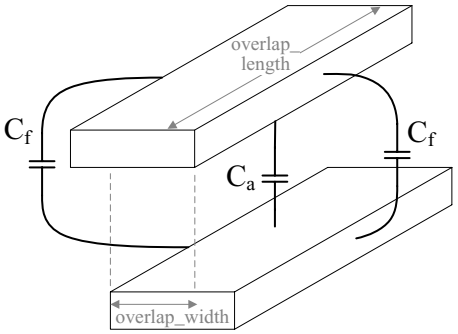


Fig. 7.10 2.5-D Capacitance between two conductors on different layers $C_{2.5\text{-}D}$ where the conductors partially or completely overlap: C_a and C_f components

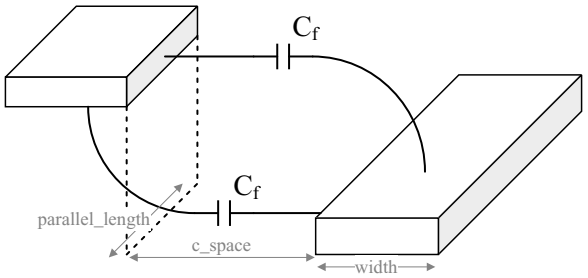


Fig. 7.11 2.5-D Capacitance between two conductors on different layers $C_{2.5\text{-}D}$ with non-overlapping conductors: only C_f component

where C_c is the coupling capacitive component obtained by the interpolation in conductor's *c_space*, and *parallel_length* is the length on which the two conductors laterally overlap. The C_c value for distant parallel or narrow perpendicular conductors tends to zero.

7.3.2.3 2.5-D Capacitance

The 2.5-D term it is used to describe the situations not encompassed by the substrate and lateral capacitances. It is computed between two conductors (shapes of a device's terminal or interconnect) on different layers, as presented in Fig. 7.10. However, there are two distinct situations that need to be considered: when the conductor plates partially or completely overlap, i.e., both area and fringe component are present, as in Fig. 7.10; and, when the conductor plates do not overlap, i.e., only fringe components, as illustrated in Fig. 7.11. For the first situation, the coupling capacitance $C_{coupling}$ depends on the C_a and C_f components between the two conductors, as indicated by Eq. (7.4):

$$C_{2.5-D} = \left(C_a\left(overlap\_width\right) \times overlap\_length\right) + \left(2 \times Cf\left(space\right) \times overlap\_length\right) \tag{7.4}$$

where C_a is interpolated in the amount of overlap, i.e., *overlap_width*. If the amount of overlap increases, i.e., larger C_a value, the $C_{2.5\text{-}D}$ also increases. Again, the C_f component is interpolated with the *space* between the conductor plate above and the closest shape in the same layer.

For the second situation, of Fig. 7.11, the C_a component is removed from Eq. (7.4) and the $C_{2.5\text{-}D}$ is defined as:

$$C_{2.5-D} = 2 \times Cf\left(c\_space\right) \times \delta \times parallel\_length \tag{7.5}$$

where C_f is interpolated in the 2-dimensional space between the two conductors, i.e., *c_space*, and computed over the *parallel_length*. As observed in Fig. 7.2, the empirical data provided by the foundry considers infinite plates above and below the reference conductor, however, this special situation of non-overlapping areas requires the introduction of a decay factor δ to reduce the ever-increasing C_f component with *c_space*. The decay factor δ was experimentally obtained with sampling on a commercial parasitic extractor for several values of *c_space*, the model is approximately linear with the increase of *c_space*, however, dependent of the pairs of conductors being considered.

7.3.3 Geometrical Considerations

In Fig. 7.12b–d, the transistor MOSFET from the generic floorplan of Fig. 7.12a, illustrates the shapes considered to compute the intra device capacitances, using the previous extraction methods. In Fig. 7.12e the parasitic capacitance between terminals of different devices, and, in Fig. 7.12f, the parasitic capacitance between a device terminal and a path of the global routing. In all the cases, the total parasitic capacitance is the sum of all partial parasitic components.

In the scenario of Fig. 7.12f, the proposed approach transforms all global routing paths into rectangular shapes prior to the parasitic estimation. In standard parasitic extraction tools this approach is impossible, due to the presence of design rule check (DRC) and layout versus schematic (LVS) errors, however, in AIDA-L the routing shapes are associated to the corresponding wire/net, and hence it is possible to detect unwanted overlaps. In the case of minimum distance violation after the expansion of the global routing path, the minimum value allowed by the technology process is considered instead for the *c_space* variable. In the case of illegal overlaps between routing paths of different nets, which would correspond to a short-circuit in the final layout, the capacitance is computed assuming that one shape in the first metal available and other in one metal level above.

7.4 Case Study: Single Ended Two-Stage Amplifier

To assess the precision of the proposed Parasitic Extractor the single ended two-stage amplifier of Fig. 7.13 is used. The circuit was designed for the UMC 130 nm CMOS technology and two different designs are considered: the first, with 51 dB gain and

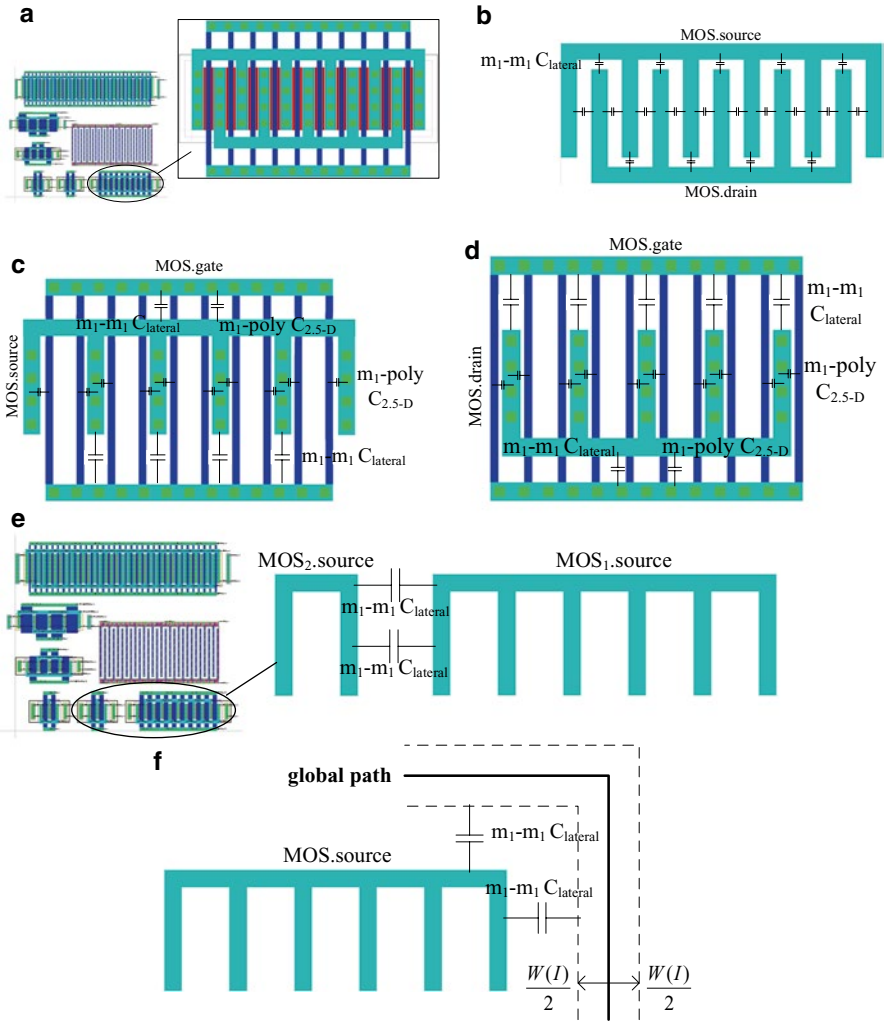


Fig. 7.12 Extracted capacitors and shapes considered: (**a**) Highlight of the transistor MOSFET from the floorplan. Transistor's shapes considered when computing the parasitic capacitance: (**b**) drain-source Cds; (**c**) gate-source Cgs; (**d**) gate-drain Cgd; (**e**) between terminals of different devices; and (**f**) between a device's terminal and a path of the global routing. The total parasitic capacitance is the sum of all partial parasitic component, not all the components are illustrated

a layout area of 310 μm^2; and, a second design, with 75 dB gain and a layout area of 1330 μm^2. The corresponding layouts generated with AIDA-L are illustrated in Figs. 7.14 and 7.15, respectively.

In Table 7.1 the results for the direct comparison between AIDA-L's Parasitic Extractor and Mentor Graphics' Calibre® for the bulk capacitance of each net for both designs are presented. The bulk capacitances are extracted with AIDA-L's

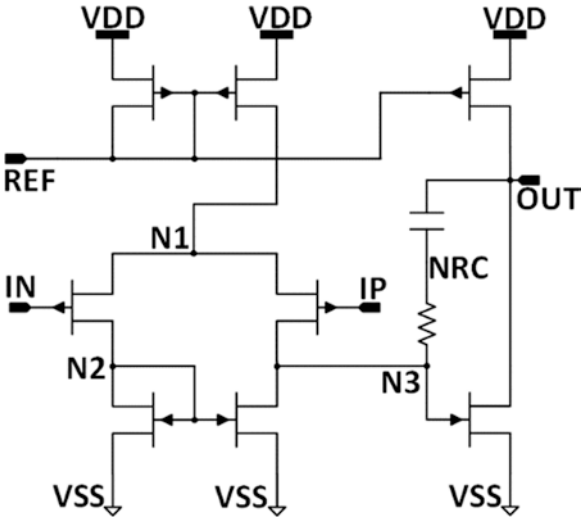


Fig. 7.13 Single ended two-stage amplifier schematic

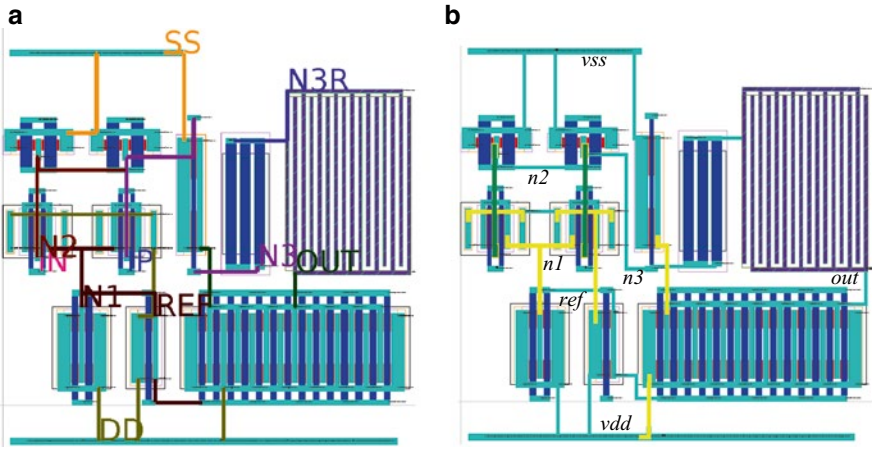


Fig. 7.14 51 dB Design: (**a**) Layout with only global routing for the 51 dB solution; (**b**) Layout with detailed routing (checks DRC and LVS). Layout area: 310 μm^2

Parasitic Extractor over the simplified layouts of Figs. 7.14a and 7.15a, and with Calibre® over the detailed layouts of Figs. 7.14b and 7.15b. Only 75 dB design presents considerable discrepancies (underestimations) in capacitance comparison due to the huge dimensions of the layout, which is more than 4 times larger than the layout area of the 51 dB design, and the detailed paths have considerable discrepancies from the global routing paths (as a result from the optimization-based detailed routing). However, the values extracted from the global routing, while underestimated, follow the ones from the manufacturable layout.

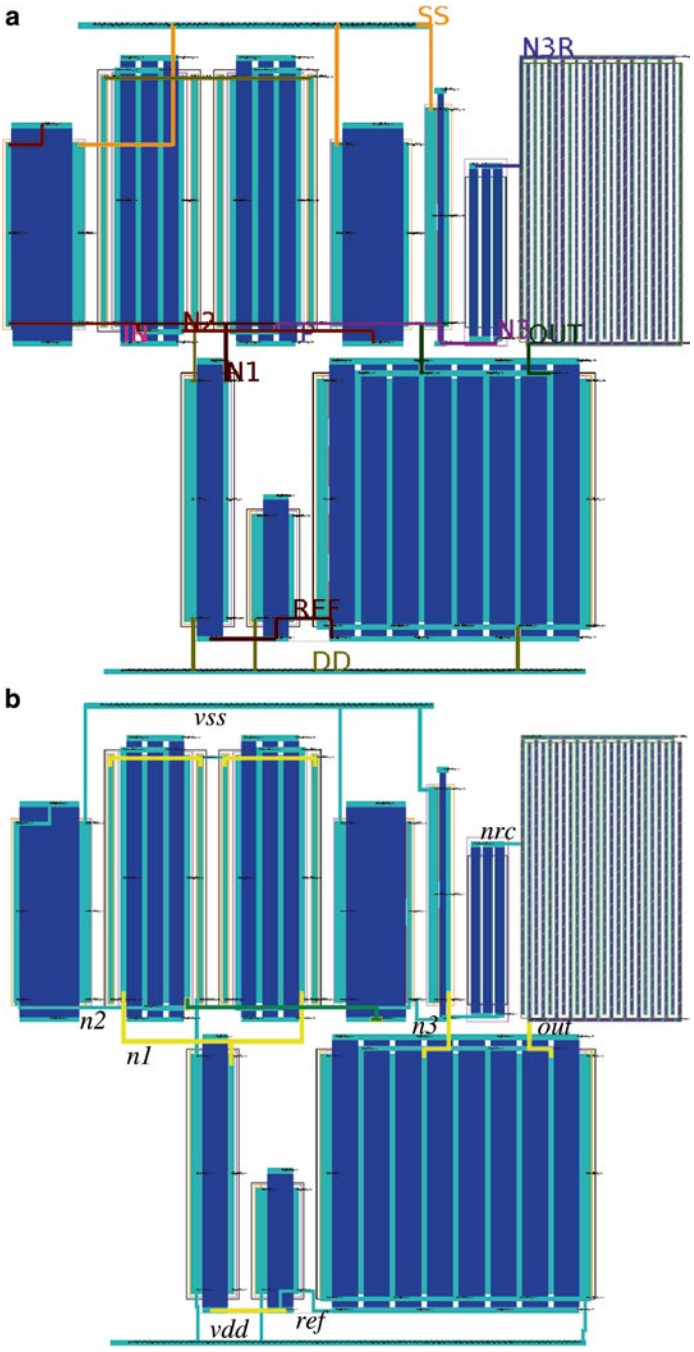


Fig. 7.15 75 dB Design: (**a**) Layout with only global routing; (**b**) Layout with detailed routing (checks DRC and LVS). Layout area: 1330 μm². Not in the same scale of layouts of Fig. 7.14

Table 7.1 Direct bulk capacitance comparison for each net of the circuit between AIDA-L's Parasitic Extractor and Mentor Graphics' Calibre®

	NET	REF	N2	IP	IN	N3	NRC	N1	OUT
51 dB	AIDA-L Parasitic Extractor[a] (fF)	6.945	1.873	0.455	0.436	1.200	1.477	0.638	1.716
	Calibre®[b] (fF)	7.115	1.713	0.389	0.389	0.903	1.129	0.581	1.884
	Discrepancy (fF)	–0.170	0.159	0.065	0.047	0.297	0.3477	0.056	–0.168
75 dB	AIDA-L Parasitic Extractor[c] (fF)	15.956	5.262	3.391	3.394	2.444	1.906	2.145	3.103
	Calibre®[d] (fF)	19.864	8.709	3.610	3.610	4.703	1.561	6.841	2.814
	Discrepancy (fF)	–3.908	–3.447	–0.219	–0.216	–2.259	0.345	–4.696	0.289

[a]Extracted over the floorplan with global routing of Fig. 7.14a
[b]Extracted over the floorplan with detailed routing of Fig. 7.14b
[c]Extracted over the floorplan with global routing of Fig. 7.15a
[d]Extracted over the floorplan with detailed routing of Fig. 7.15b

In Tables 7.2 and 7.3 the same comparison is performed for the interconnect capacitances net-to-net.

Again, the only considerable discrepancies (underestimations) appear in the 75 dB design, and the larger errors occur on the capacitances to the power nets, i.e., VDD-N1, VDD-N2, VDD-N3 and VSS-N2. While overlooked in the simplified extraction procedure, the impact of the substrate/well biasing regions included in the PMOS/NMOS transistors is relevant to the amount of parasitics structures identified in the detailed layout. However, the proposed Parasitic Extractor is still slightly conservative, pointing for underestimation instead of overestimation. A slightly underestimation of the layout parasitics should still affect the circuits' post-layout performance in the same direction, while overestimation could compromise the convergence of the layout-aware circuit sizing optimization for harsh specifications.

With the complete list of extracted parasitic structures, the nets in the original netlist are updated in AIDA-C to account for the parasitic effects, where each wire is modeled as a lumped transmission line using the $\pi 2$ model [11]. By iteratively introducing parasitic structures extracted with AIDA-L and observing the circuits' performance it was verified that intra-device parasitics, commonly unnecessarily overlooked in the pre-layout simulation despite being easy to include accurately in the models by fixing the device-level layout design (as done in RF models), can have a significant impact in the circuit performance. However, for the accurate evaluation of the circuits' performance in the presence of parasitics, the wires contributions are indispensable.

Table 7.2 Direct interconnect capacitance comparison between AIDA-L's Parasitic Extractor and Mentor Graphics' Calibre®, for the 51 dB design

		Mentor Graphics' Calibre®[a] (fF)									
	NET	REF	N2	IP	IN	N3	NRC	N1	VDD	VSS	OUT
AIDA-L Parasitic Extractor[b] (fF)	REF		0	0.030	0	0.238	0	0.355	3.414	0	2.372
	N2	0.009		0.041	0.2799	0.252	0	0.465	0.114	0.899	0
	IP	0.026	0.097		0	0.222	0	0.261	0.052	0.001	0
	IN	0	0.278	0.009		0	0	0.297	0.026	0	0
	N3	0.172	0.266	0.200	0		0.011	0.486	0.229	0.722	0.268
	NRC	0.002	0	0	0	0.018		0	0	0.002	0.021
	N1	0.359	0.449	0.224	0.224	0.384	0		1.216	0	0
	VDD	3.603	0.001	0.005	0	0.018	0.070	0.460		0.006	3.143
	VSS	0.003	0.802	0.031	0.136	0.503	0	0.010	0.002		0.424
	OUT	3.044	0	0	0	0.412	0.036	0.015	3.397	0.421	

[a]Extracted over the floorplan with detailed routing of Fig. 7.14b
[b]Extracted over the floorplan with global routing of Fig. 7.14a

Table 7.3 Direct interconnect capacitance comparison between AIDA-L's Parasitic Extractor and Mentor Graphics' Calibre®, for the 75 dB design

		Mentor Graphics' Calibre®[a] (fF)									
	NET	REF	N2	IP	IN	N3	NRC	N1	VDD	VSS	OUT
AIDA-L Parasitic Extractor[b] (fF)	REF		0.195	0	0	0.054	0.046	0.917	9.596	0.002	7.196
	N2	0.464		0.155	2.255	1.363	0.006	0.892	1.159	1.928	0.003
	IP	0.003	0.152		0	2.249	0	2.196	0.176	0.017	0
	IN	0	2.547	0.029		0	0	2.209	0.191	0.018	0
	N3	0.091	1.305	2.511	0		0.175	0.747	0.916	0.805	0.729
	NRC	0.001	0.008	0	0	0.085		0	0	0	0.001
	N1	0.905	2.098	2.585	2.562	1.640	0		2.159	0.011	0
	VDD	9.407	0.011	0	0	0.004	0.049	0.384		0.594	0.908
	VSS	0.022	1.380	0	0.028	0.775	0.013	0	0		0.911
	OUT	7.938	0. 003	0	0	0.819	0.288	0.008	2.143	1.045	

[a]Extracted over the floorplan with detailed routing of Fig. 7.15b

[b]Extracted over the floorplan with global routing of Fig. 7.15a

Table 7.4 Execution time for each task implemented in AIDA-L for the single ended two-stage amplifier

	AIDA-L			
	Template-based Placer (s)	Global routing[a] (s)	Detailed routing (s)	Parasitic Extractor (s)
51 dB Design	<0.01	~0.01	~70	~0.2
75 dB Design	<0.01	~0.01	~110	~0.3

Computational times obtained on a Fedora Virtual Machine running on Intel® Core™ i7-3770 @ 3.4 GHz with 32 GB of RAM

[a]Global routing times encompasses: Wiring planner, Symmetry planner and Global routing

Although the discrepancies found in Tables 7.1, 7.2, and 7.3 for the direct parasitic capacitance comparison between global and detailed routing solutions, the case studies addressed later in Chap. 8 of this book confirm that the impact in the performances of the circuits are similar, and with a precision sufficient to guide the parasitic-aware optimization in the right direction.

7.5 Conclusion

As overviewed in Chap. 2 of this book, most layout-aware approaches where parasitic accuracy is high require the detailed layout generation at each iteration, and for that reason, rely on procedural generators that lack the flexibility to handle change, i.e., changing the topology can discard most or all the previous setup work. Traditionally, the last step required in the automatic layout generation flow is the detailed routing, which is one of the most time consuming tasks on the automatic analog layout flow. The proposed Parasitic Extractor implements straightforward but accurate models for the technology-dependent parameters, and, a complete 2.5-D parasitic capacitances extraction over the floorplan and early-stages of routing.

The computational time of each task implemented in AIDA-L for the two overviewed designs is presented in Table 7.4. The precision lost when using the lightweight AIDA's Parasitic Extractor [12] when comparing to Calibre® (or other commercial extractor by analogy), is not only rewarded by the small extraction time required for this simplified extractor (and with an unlimited parallelization value), but specially by the fact that the procedure operates over the global routing approximation, avoiding the need of the final detailed layout during the sizing optimization iterations. Still, the precision is sufficient to guide the parasitic-aware optimization in the right direction. For example, in the 51 dB case, the inclusion of AIDA-L's detailed routing in the *parasitic-aware* loop would degrade the computational times in 70 s for each point evaluated. For a layout-aware circuit sizing with 128 points optimized through 1000 generations, will result in months of computational effort.

References

1. P. Vancorenland, G. Van der Plas, M. Steyaert, G. Gielen, W. Sansen, A layout-aware synthesis methodology for RF circuits, in *Proc. IEEE/ACM ICCAD*, 2001, pp. 358–362
2. A. Pradhan, R. Vemuri, Efficient synthesis of a uniformly spread layout aware pareto surface for analog circuits, in *International Conference on VLSI Design*, Jan 2009, pp. 131–136
3. Y.-C. Liao, Y.-L. Chen, X.-T. Cai, C.-N.J. Liu, T.-C. Chen, LASER: Layout-aware analog synthesis environment on laker, in *GLSVLSI'13*, May 2013, pp. 107–112
4. S. Youssef, F. Javid, D. Dupuis, R. Iskander, M.-M. Louerat, A python-based layout-aware analog design methodology for nanometric technologies, in *IEEE 6th Int. IDT*, Dec 2011, pp. 62–67
5. M. Ranjan et al., Fast, layout-inclusive analog circuit synthesis using pre-compiled parasitic-aware symbolic performance models, in *DATE Conference and Exhibition*, Feb 2004, pp. 604–609
6. H. Habal, H. Graeb, Constraint-based layout-driven sizing of analog circuits. IEEE TCAD **30**(8), 1089–1102 (2011)
7. R. Castro-Lopez, O. Guerra, E. Roca, F. Fernandez, An integrated layout-synthesis approach for analog ICs. IEEE TCAD **27**(7), 1179–1189 (2008)
8. A. Toro-Frias, R. Castro-Lopez, F. Fernandez, An automated layout-aware design flow, in *Int. Conf. SMACD*, Sept 2012, pp. 73–76
9. G. Berkol, A. Unutulmaz, E. Afacan, G. Dundar, F.V. Fernandez, A.E. Pusane, F. Baskaya, A two-step layout-in-the-loop design automation tool, in *IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, June 2015, pp. 1–4
10. W.H. Kao et al., Parasitic extraction: Current state of the art and future trends, in *Proceedings of the IEEE*, May 2001, pp. 729–739
11. J. Gong, D.Z. Pan, P.V. Srinivas, Improved crosstalk modeling for noise constrained interconnect optimization, in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2001, pp. 373–378
12. B. Cardoso, R. Martins, N. Lourenço, N. Horta, AIDA-PEx: Accurate parasitic extraction for layout-aware analog integrated circuit sizing, in *11th Conference on PhD Research in Microelectronics and Electronics (PRIME)*, June 2015
13. G. Shomalnasab, H. Heys, L. Zhang, Analytic modeling of interconnect capacitance in submicron and nanometer technologies, in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2013, pp. 2553–2556

Chapter 8
Experimental Results

This chapter presents and discusses the application of the proposed, and implemented, methodologies to practical analog integrated circuit (IC) layout design examples. The framework of the proposed methodology for the automatic generation of analog ICs layout is coded in JAVA™, and integrated in the AIDA's framework (Martins et al., *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2012, pp. 29–32; Lourenço et al., *Design, Automation & Test in Europe Conference (DATE)*, 2015, pp. 1156–1161; Martins et al., *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2015, pp. 1–4). In order to use the implemented platform, the first design task was the development of the technology design kit. Since the presented United Microelectronics Corporation (UMC) 130 nm design kit and the AIDA analog module generator accompanied the development of the tool, it is difficult to establish a development time. However, the development of a similar design kit, e.g., 350, 180 or 130 nm from other foundry can be achieved in less than 1 week.

8.1 Organization of the Results

As seen thus far, AIDA-L implements a comprehensive set of features, and not all of them are applied to each presented test case [1–3]. To ease tracking the techniques used in each design example, Table 8.1 maps each case study, circuit or benchmark, to the methodologies being applied. During the development of the proposed methodologies several analog cells and benchmark sets were used to test and verify the methodologies, however, only seven different test cases were selected and presented for this document. First, four analog cells, namely, a single stage amplifier using voltage combiner, a single ended two-stage amplifier, a two-stage folded cascode amplifier and an operational transconductance amplifier (OTA) are used to demonstrate the complete layout generation flow; and then, three benchmark sets are used, two to evaluate the innovative optimization-based Placer and one for the fully-automatic Router.

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_8

Table 8.1 Organization of the experimental results chapter

	Section	Template-based Placer (Chap. 4)	Optimization-based Placer (Chap. 5)		Fully-automatic Router (Chap. 6)				Parasitic Extractor (Chap. 7)	Post-layout validation
			Hierarchical opt. in absolute coordinates	Current-flow and current-density considerations	Electromigration-aware wiring topology	Symmetry Planner	Global Router	Detailed Router		
Single stage amplifier using voltage combiner	8.2	✓	✓	✓	✓	✓	✓	✓		✓
Single ended two-stage	8.3	✓			✓	✓	✓	✓	✓	✓
Two-stage folded cascode	8.4	✓	✓	✓	✓	✓	✓	✓	✓	✓
OTA	8.5	✓			✓		✓	✓		
Placement benchmark	8.6.1		✓							
MNCN benchmarks	8.6.2		✓							
Routing benchmark	8.6.3				✓		✓			

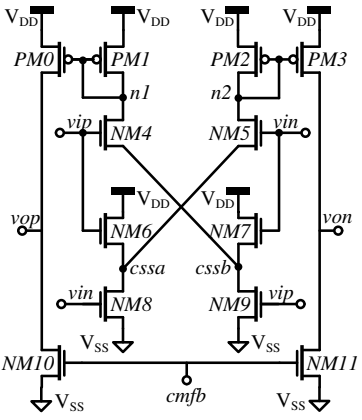
8.2 Case Study I: Single Stage Amplifier with Gain Enhancement Using Voltage Combiner

The proposed approach is first used to automate the layout design of an amplifier topology using voltage combiners proposed in [14], for the United Microelectronics Corporation (UMC) 130 nm design process. The schematic of the circuit is presented in Fig. 8.1a and the target specifications in Fig. 8.1b. The methodologies applied to this circuit are outlined in Table 8.2.

8.2.1 Inputs: Template File Definition and Floorplan-Aware Circuit Sizing

The circuit was optimized using AIDA-L's template-based Placer, integrated with AIDA-C, to perform a floorplan-aware circuit sizing, as overviewed in Chap. 3 of this book. The corner conditions (fixed temperature of 25 °C and supply voltage of 3.3 V; and the devices' models: slow NMOS and slow PMOS, slow NMOS and fast PMOS, fast NMOS and slow PMOS and, fast NMOS and fast PMOS) were considered while optimizing the circuit sizing, whose objectives were set to minimize the real floorplan area (not estimated) and maximize the low-frequency gain (GDC) for

a



b

Design Specifications	
area [µm²]	min
FOM [MHz*pF/mA]	≥ 1000
IDD [µA]	≤ 350
GDC [dB]	max ≥ 50
GBW [MHz]	≥ 30
PM [º]	≥ 60

Fig. 8.1 Single stage amplifier with gain enhancement: (**a**) schematic and (**b**) specifications [14]

Table 8.2 Methodologies applied to the single stage amplifier

	Template-based Placer (Chap. 4)	Optimization-based Placer (Chap. 5)	Fully-automatic Router (Chap. 6)	Post-layout validation
Section	8.2.2	8.2.3	8.2.2 8.2.3	8.2.3

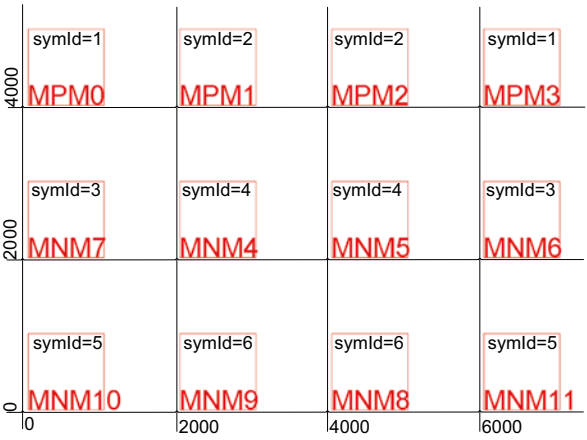


Fig. 8.2 Topological relations between cells of the template file

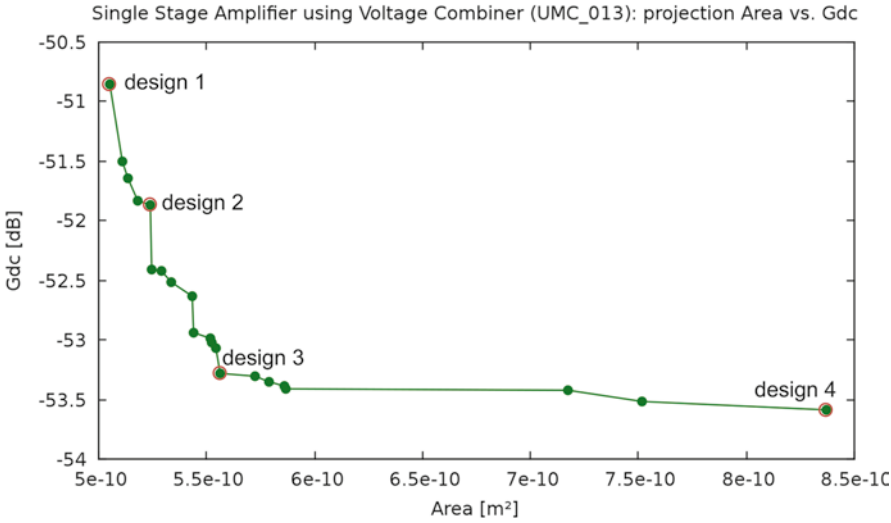


Fig. 8.3 POF representing the tradeoffs between the two optimization objectives: area (m^2) versus GDC (dB)

the worst case corner. Besides the specifications from Fig. 8.1b, the devices' functional specifications are $V_{DS} - V_{DSat} \geq 50$ mV for all devices, the PMOS overdrive $(V_{GS} - V_{TH}) \geq 100$ mV, and the NMOS ≥ 50 mV, were also set. The optimization variables were the width, length and the number of fingers of all the MOS devices. Additional details are remitted to the original design [14].

The proposed design flow starts with the definition of the template file. A straightforward template file was used during the optimization process to obtain the floorplan for each of the sizing solutions, the corresponding topological relations between cells are presented in Fig. 8.2. The result is a Pareto optimal front (POF) with 21 different sizing solutions representing the tradeoffs between the optimization objectives, which is presented in Fig. 8.3.

Table 8.3 Four different designs for the single stage amplifier with gain enhancement, for the UMC 130 nm design process

		Design 1	Design 2	Design 3	Design 4
Specifications/performances for the worst case corner					
Area (μm^2)	min	504.5	523.7	556.3	837.2
FOM (MHz*pF/mA)	≥1000	1068	1051	1073	1068
IDD (μA)	≤350	347.6	335.5	349.9	34.75
GDC (dB)	max ≥ 50	50.855	51.869	53.281	53.592
GBW (MHz)	≥30	42.39	39.88	42.69	41.83
PM (°)	≥60	69.77	69.61	64.62	60.25
Devices' sizes					
MPM_0 MPM_3	w0 (μm)	23	26.5	22.1	38.6
	l0 (nm)	570	590	670	720
	n0	4	4	4	2
MPM_1 MPM_2	w1 (μm)	1.8	2.1	1.4	2.4
	l1 (nm)	300	300	300	300
	n1	2	2	2	2
MNM_6 MNM_7	w6 (μm)	48.6	47.1	50.9	60.2
	l6 (nm)	610	590	610	730
	n6	8	8	8	8
MNM_4 MNM_5	w4 (μm)	11	10.9	10.6	16.2
	l4 (nm)	610	570	570	710
	n4	8	6	6	4
MNM_8 MNM_9	w8 (μm)	1	1	1	1
	l8 (nm)	940	930	940	940
	n8	2	2	2	2
MNM_{10} MNM_{11}	w10 (μm)	8.1	2.8	5	4.9
	l10 (nm)	680	890	940	900
	n10	2	2	2	2

8.2.2 *Layout Generation: Template-Based Placer*

Four different sizing solutions were selected from the POF, whose performances and devices' sizes are presented in Table 8.3. The layout of those four sizing solutions was automatically generated using the proposed layout generation methodology (template-based Placer and complete routing) and is presented in Figs. 8.4, 8.5, 8.6, and 8.7, design 1, design 2, design 3 and design 4, respectively, while the execution times are presented in Table 8.4.

The circuit contains 11 different nets that need to be routed, each one connecting from a range of 2–7 multiport (MP) terminals. The inputs of the Router are the floorplan and the set of electric-currents obtained automatically (specific to each different solution) and the netlist of the circuit (which is common to all the designs). No additional setup or tuning is required to generate the electromigration (EM)-aware and symmetric final routing. Regardless the changes to the sizes of the devices or to the relative positioning of the cells, there is absolutely no additional setup

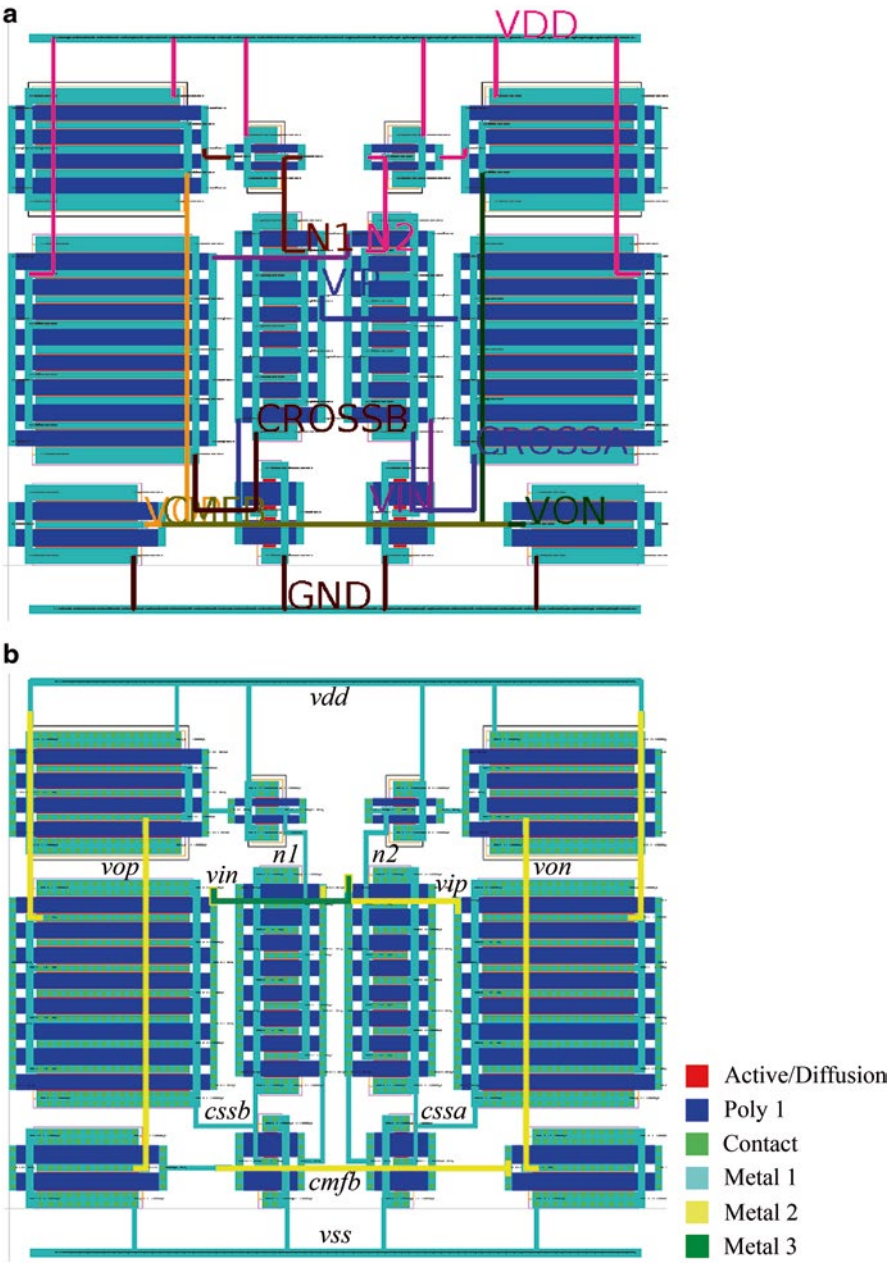


Fig. 8.4 Automatically generated layout for design 1 (50.855 dB; 504.5 μm^2) (UMC 130 nm): (**a**) only global routing; (**b**) detailed layout. Drawn in the same scale of Figs. 8.5 and 8.6 layouts

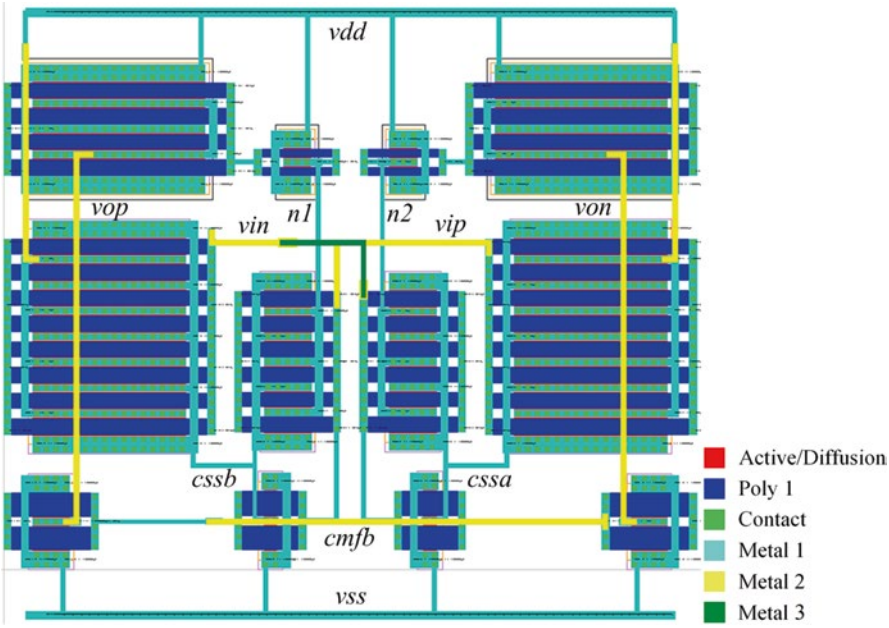


Fig. 8.5 Automatically generated detailed layout for design 2 (51.869 dB; 523.7 μm²) (UMC 130 nm). Drawn in the same scale of Figs. 8.4 and 8.6 layouts

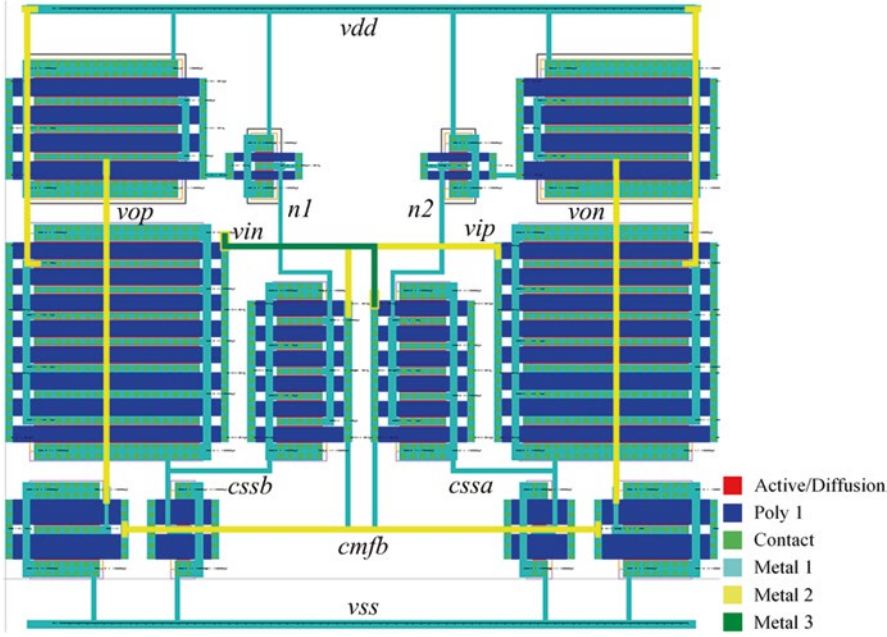


Fig. 8.6 Automatically generated detailed layout for design 3 (53.281 dB; 556.3 μm²) (UMC 130 nm). Drawn in the same scale of Figs. 8.4 and 8.5 layouts

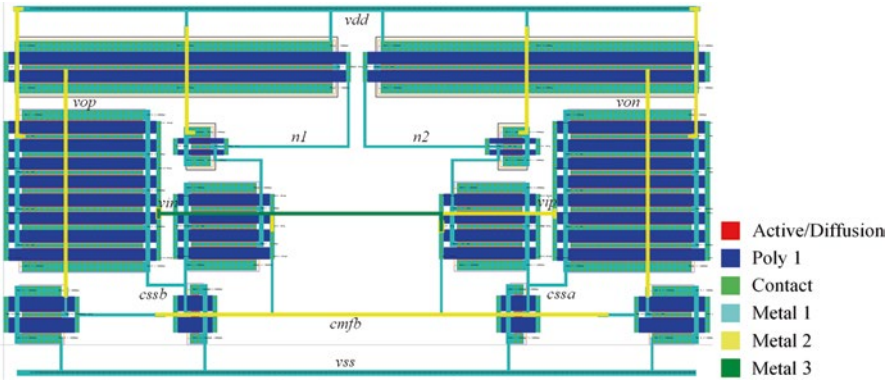


Fig. 8.7 Automatically generated detailed layout for design 4 (53.592 dB; 837.2 μm^2) (UMC 130 nm). Drawn in a different scale of Figs. 8.4, 8.5 and 8.6 layouts

Table 8.4 Execution time for each task implemented in AIDA-L for the single stage amplifier

	AIDA-L		
	Template-based Placer (s)	Global routing[a,b] (s)	Detailed routing[b] (s)
Design 1, design 2, design 3 and design 4	<0.01	~0.01	~40

Computational times obtained on an Intel® Core™ i7-3770 @ 3.4 GHz with 32 GB of RAM
[a]Global routing times encompasses: Wiring planner, Symmetry planner and Global routing
[b]Computational time to route each different floorplan solution. The maximum number of available cores in the machine were used to perform the built-in layout evaluation procedures of each population, at each generation

effort to route any point of the POF. Also, power nets for *vdd* and *gnd* potential are placed above and/or below the circuit automatically. While the electric-currents at each terminal were relevant to sketch an EM-aware wiring topology for each net, since the electric-current values are low they are not reflected in the interconnects' widths for any net of the four designs. The minimum width allowed by the technology process is used in all the wires.

8.2.3 Layout Generation: Optimization-Based Placer with Current-Flow and Current-Density Considerations

In the previous sections, the amplifier's floorplan was designed following designer's guidelines from the template. In this section, it is optimized using the optimization-based Placer and most of the information contained in the template file is discarded (only symmetry and matching information is kept). The circuit is optimized for the devices' sizes of design 1 of previous section, constrained to the current-paths presented in Table 8.5, i.e., paths from power to ground lines. As previously introduced

Table 8.5 Current-flow constraints for the single stage amplifier using voltage combiner, i.e., paths from power to ground lines

Current-flow constraints
$V_{dd} \rightarrow \mathrm{PM}_0 \rightarrow \mathrm{NM}_{10} \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{PM}_1 \rightarrow \mathrm{NM}_4 \rightarrow \mathrm{NM}_9 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{NM}_6 \rightarrow \mathrm{NM}_8 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{PM}_2 \rightarrow \mathrm{NM}_5 \rightarrow \mathrm{NM}_8 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{NM}_7 \rightarrow \mathrm{NM}_9 \rightarrow V_{ss}$
$V_{dd} \rightarrow \mathrm{PM}_3 \rightarrow \mathrm{NM}_{11} \rightarrow V_{ss}$

in Chap. 5 of this book, the optimization of the amplifier was divided into two partitions, one containing the PMOS, and the other the NMOS devices.

The optimization of the top partition resulted in a front with 50 non-dominated placement solutions, all verifying the current-flow constraints, from the exploration of the tradeoff placement's *width*, placement's *height* and sum of the optimal EM-aware wiring topologies (*WTs*). The projections for each pair of objectives are presented in Fig. 8.8. The two floorplan solutions that present the lowest objective values, i.e., the solution that minimizes both *width* and *WT*, and, the solution that minimizes *height*, are also highlighted in the projections and the corresponding routed layout is presented in Fig. 8.8a, b. Additionally, a floorplan optimization without current-flow and current-density considerations was performed, and the obtained layout, after routing, is presented in Fig. 8.8c.

The automatically generated layouts were design-rule check (DRC) and layout-versus-schematic (LVS) verified using commercial tools, and their post-layout simulation results are found in Table 8.6. The post-layout simulation of the layout obtained using the template-based Placer is also shown. The relevant computational times are presented on Table 8.7.

While layout 2 fulfils all the current-flow constraints, layout 1 presents the performance values closer to nominal due to the smaller *WT* and consequently smaller interconnect length and routing congestion. This reinforces the fact that fulfilling the current-flow constraints is not enough to guarantee that a floorplan with a desirable aspect ratio is obtained, and consequently, layout 2 has the worst WT and routing area. That fact is reflected negatively in some post-layout performances, especially offset voltage and phase margin that are borderline with the specifications.

Layout 3 outperforms previous designs in terms of placement area (and layout 2 in *WT*), however, due to the lack of current-flow considerations, which increased the routing-induced parasitics, the offset voltage falls way beyond the specification, even though, the total wire length is smaller than in layout 2. Layout 2 by its part, it is more sparse, favoring a reduction on routing-induced parasitics. By analyzing layout 3, the unfulfilled current-flow constraints, depicted in Fig. 8.9, led to more complicated routing topologies. While the inexistence of *WT* minimization led to longer routing paths even in this compact floorplan. This increase in the interconnect length is representative of higher interconnect resistance that further impacted offset voltage.

The template-based floorplan, which inherits valuable information from the designer's expertise, e.g., aligned current/signal-flows, minimal WT, minimization of parasitics, etc., outperforms all other layouts in terms of offset voltage. Nevertheless, layout 1 shows very competitive results, showing the importance of including current-flow and current-density considerations for an effective fully-automatic placement in the absence of careful, albeit time-consuming, handmade guidelines.

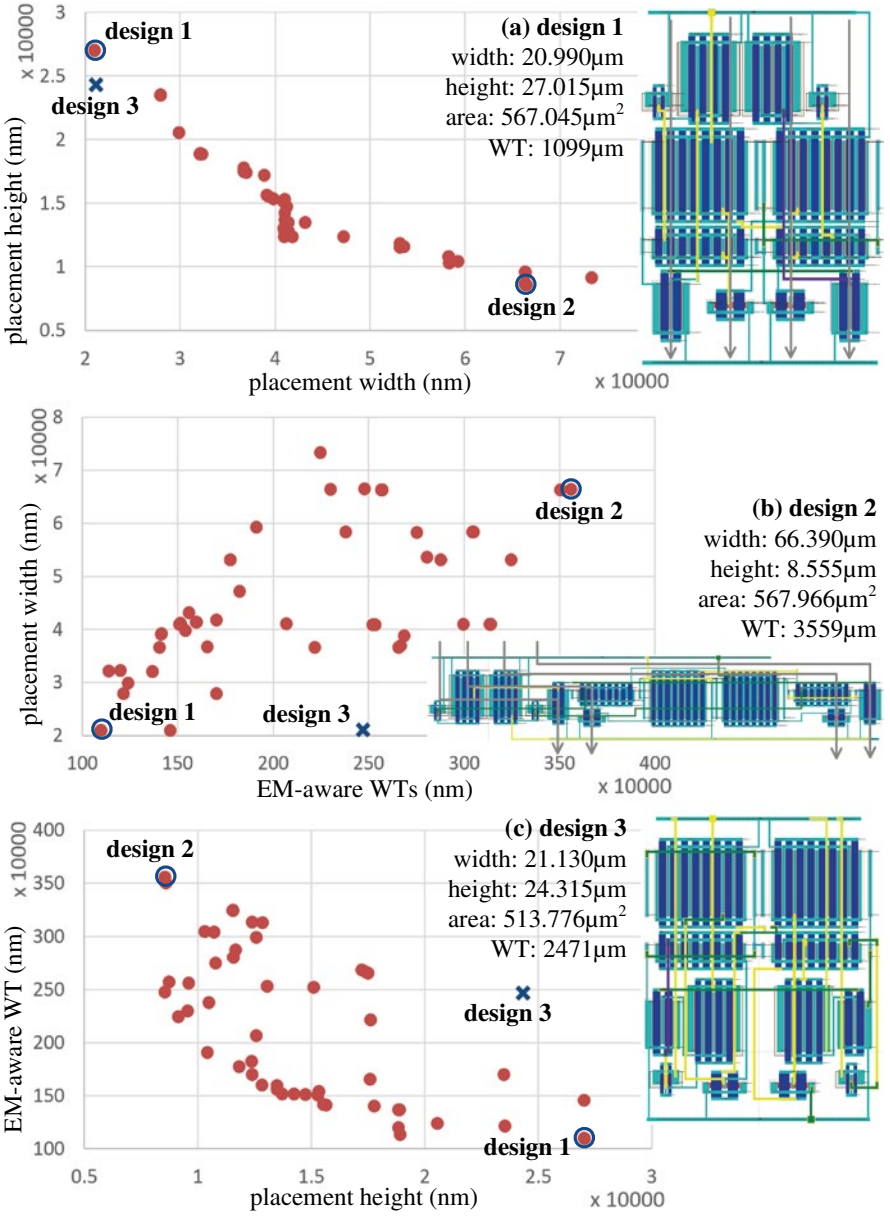


Fig. 8.8 Placement optimization of the single stage amplifier with current-flow constraints for the tradeoff placement's *width*, *height* and sum of the optimal EM-aware WTs. Different projections of the 50 different placement solutions found. Layouts are not drawn at the same scale [15]

Table 8.6 Post-layout performances for single stage amplifier using voltage combiner

Performance	Spec.	Netlist (nominal)	Post-layout[a]			
			Layout 1	Layout 2	Layout 3[b]	Template-based floorplan
GBW (MHz)	≥30	53.196	52.644	52.525	52.478	52.670
GDC (dB)	≥50	51.238	51.226	51.215	51.212	51.219
IDD (μA)	≤5e^{-4}	2.846e^{-4}	2.847e^{-4}	2.848e^{-4}	2.848e^{-4}	2.847e^{-4}
PM (°)	≥70	71.833	70.277	69.914	70.072	70.382
VOFF (mV)	≤5	1.234e^{-11}	2.329	4.976	83.599	0.804
FOM (MHz*pF/mA)	≥1100	1121.5	1109.5	1106.8	1105.8	1109.9

[a]DRC, LVS and extraction performed in Mentor Graphics' Calibre® [5]
[b]Solution obtained with traditional optimization without current-flow/current-density considerations

Table 8.7 Post-layout performances for single stage amplifier using voltage combiner

	AIDA-L		
	Placer (s)	Global routing[a,b] (s)	Detailed routing[b] (s)
Optimization-based with current-flow and current-density	~130	~0.01	~40
Optimization-based without current considerations	~2		
Template-based	<0.01		

Computational times obtained on an Intel® Core™ i7-3770 @ 3.4 GHz with 32 GB of RAM
[a]Global routing times encompasses: Wiring planner, Symmetry planner and Global routing
[b]Computational time to route each different floorplan solution. The maximum number of available cores in the machine were used to perform the built-in layout evaluation procedures of each population, at each generation

8.3 Case Study II: Single Ended Two-Stage Amplifier

In this Section the developed methodologies outlined in Fig. 8.10a are applied to a second circuit example, the single-ended two-stage amplifier whose schematic is presented in Fig. 8.10b.

8.3.1 Inputs: Template File Definition and Floorplan-Aware Circuit Sizing

The devices' sizes and the electric-currents flowing in each terminal were obtained using AIDA-C integrated with AIDA-L's template-based Placer, and a floorplan-aware circuit sizing is performed, following the principles introduced in Chap. 3

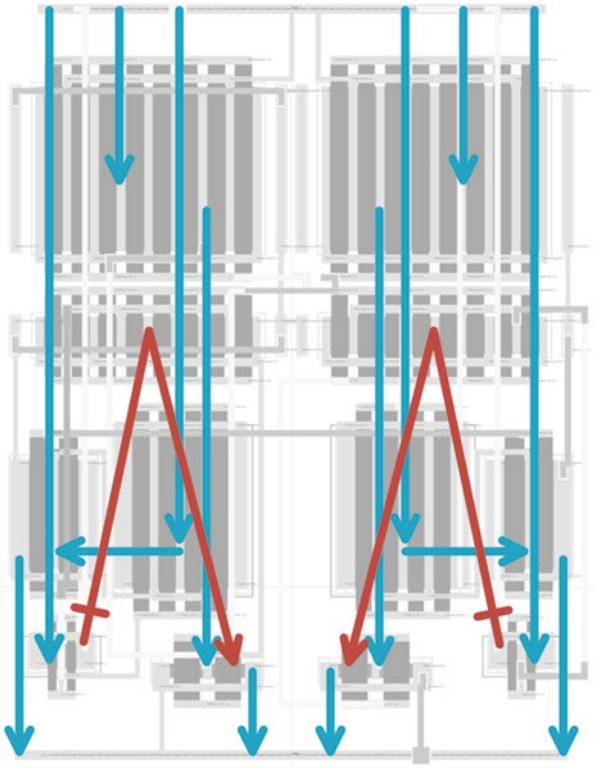


Fig. 8.9 Representation of current-flows of design 3 of Fig. 8.7 (broken current-flow constraints marked at *red*) [15]

a

	Template-based Placer (Chapter 4)	Fully-Automatic Router (Chapter 6)	Parasitic Extractor (Chapter 7)	Post-Layout Validation
Section	8.3.2 8.3.3	8.3.2 8.3.3	8.3.3	8.3.3

b

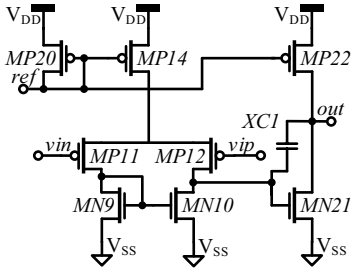


Fig. 8.10 (**a**) Methodologies applied to the single ended two-stage amplifier; (**b**) circuit schematic

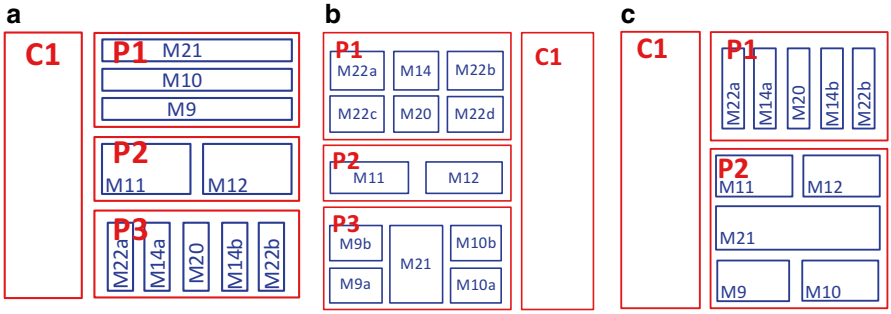


Fig. 8.11 Topological relations between cells of the different hierarchical templates (**a**–**c**)

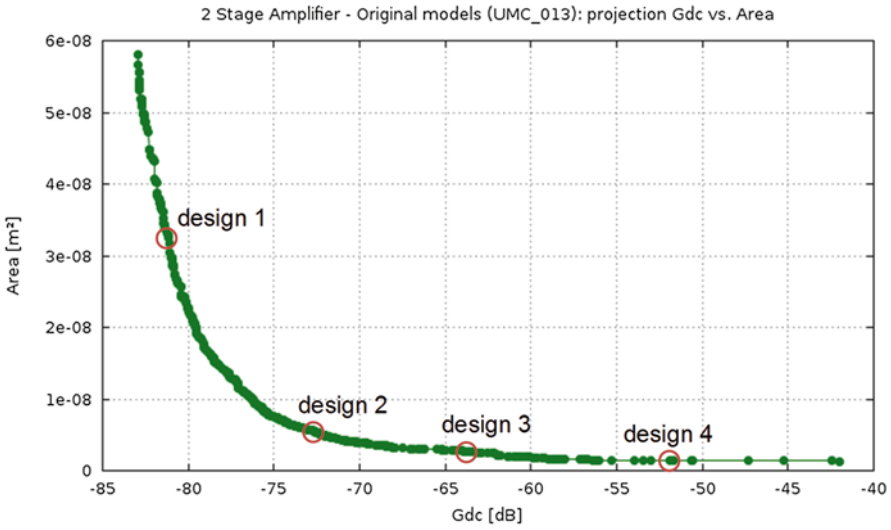


Fig. 8.12 POF representing the tradeoffs between the two optimization objectives: area (m^2) versus GDC (dB). Four different designs are highlighted

of this book. To allow better packing of the layout throughout the entire solution set, three hierarchical template files were defined and used during the floorplan-aware optimization process. The topological relations between cells of the hierarchical template files are shown in Fig. 8.11. The objectives of the circuit sizing optimization were set to maximize the GDC while minimizing the real floorplan area (not estimated). The result was a POF with 404 different sizing solutions representing the tradeoffs between the optimization objectives, which is presented in Fig. 8.12.

Table 8.8 Five different designs for the two-stage amplifier, for the UMC 130 nm design process

		Design 1	Design 2	Design 3	Design 4
Performances					
Area (μm^2)		32,537	5693	2606	1431
Gain DC (dB)		81.133	72.876	63.592	53.001
Devices' sizes					
MP_{20}	w (µm)	20	20	4.8	2
	l (nm)	400	400	200	250
	n	2	2	2	2
MP_{14}	w (µm)	20	20	4.8	2
	l (nm)	400	400	200	250
	n	2	2	2	2
MP_{11} MP_{12}	w (µm)	1459.6	274.4	120.6	57
	l (nm)	2000	900	800	500
	n	178	28	18	30
MP_{22}	w (µm)	2400	600	134.4	72
	l (nm)	400	400	200	250
	n	24	60	56	72
MN_9 MN_{10}	w (µm)	548.8	126	53	17.2
	l (nm)	5000	4750	4300	4550
	n	56	14	10	4
MN_{21}	w (µm)	1840	1160	608.4	169.2
	l (nm)	1050	300	200	120
	n	184	200	78	36
C	l (µm)	98.9	60	40.4	28.4
	nf	154	32	34	32

8.3.2 Layout Generation: Template-Based Placer

Four different sizing solutions were selected from the POF, whose devices' sizes are presented in Table 8.8, and their layout was automatically generated by AIDA-L and presented in Fig. 8.13.

The inputs to the Router are the floorplan, the set of electric-currents obtained automatically (and specific to each different solution) and the netlist of the circuit (which is common to all the designs). Again, no additional setup or tuning is required to generate the EM-aware and symmetric final routing for all the 404 different sizing solutions. The proposed routing method is not restricted to completely symmetrical/differential circuits, as demonstrated by the selected test-case. However, some transistors were split in the template files into multiple folded transistors (i.e., transistors MP_{22}, MP_{14}, MN_9 and MN_{10}) to increase the overall symmetry of the solution.

The design 4 detailed in Fig. 8.13 shows the final layout. The computational time for the template-based Placer module can be considered instantaneous. The routing

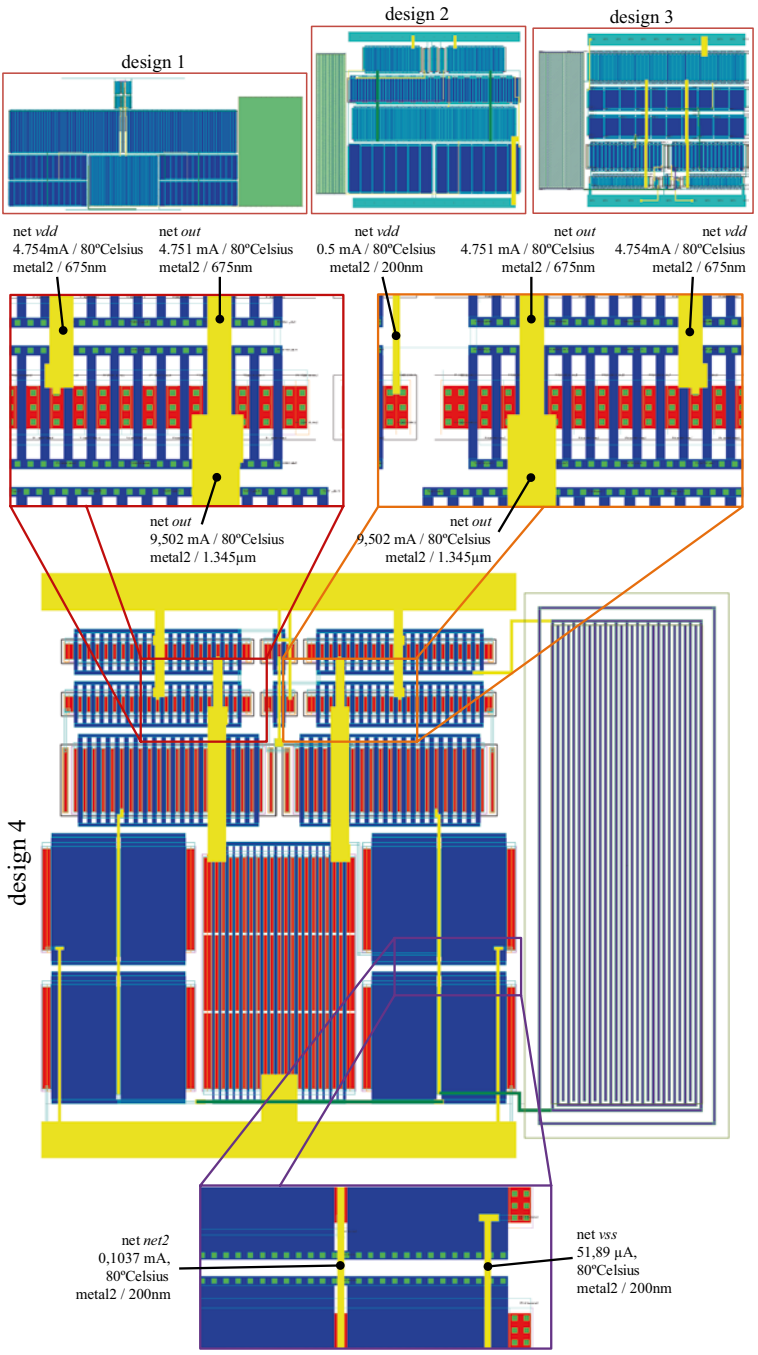


Fig. 8.13 Automatically generated layouts for the two-stage operational amplifier [GDC (dB), area (μm^2)]: design 1 (81.133; 32537) using template hierarchy (**b**); design 2 (72.876; 5693) using template hierarchy (**c**); design 3 (63.592; 2606) using template hierarchy (**a**); design 4 (53.001; 1431) using template hierarchy (**b**). Layouts are not placed in the same scale

Table 8.9 Results obtained by using the automatic routing approach in the two-stage operational amplifier (design 4)

Net	vdd	vss	ref	pair	net1	net2	out
#Terminals	11	6	7	3	7	5	6
#Ports[a]	[2–13]	[4–13]	[2–14]	[1–13]	[2–10]	[1–14]	[1–10]
#Layers/V_{cost}[b]	6/5	6/5	6/5	6/5	6/5	6/5	6/5
#Connectivity[c]	10	5	6	2	6	4	5
WT *area*[d] (nm.A)	1.957	1.795	$1.981e^{-1}$	$2.167e^{-1}$	$5.222e^{-1}$	$3.343e^{-1}$	2.317
#WS[e]	3	2	1	1	1	1	2
MP-selection *area*[f] (nm.A)	$7.941e^{-1}$	$6.504e^{-1}$	$5.747e^{-2}$	$9.562e^{-2}$	$3.717e^{-1}$	$3.027e^{-1}$	1.342
Runtime[g] (s)	0.02	<0.01	0.01	<0.01	0.01	<0.01	<0.01
Single-net detailed routing							
#Pop/#Gen[h]	128/100	128/100	128/100	128/100	128/100	128/100	128/100
Runtime (s)	9.078	6.741	4.464	5.591	17.570	5.453	21.056
Multi-net detailed routing							
#Pop/#Gen[h]	128/200						
Runtime (s)	84.350						

Computational times obtained on an Intel® Core™ i7-3610 @ 2.3 GHz with 8 GB of RAM
[a]Range of number of ports for each terminal
[b]Cost of assigning a via (for the enhanced A* search, presented on Chap. 6 of this book)
[c]Number of terminal-to-terminal wires determined in the wiring topology
[d]Wiring '*area*' of the EM-aware WT, computed by Eq. (6.6) in Chap. 6 of this book. Where l_i is the length (in nanometers) and j_i the current density (in Amperes) of the wire i. In the traditional EM-aware problem the current j_i is used instead of the wire's width due to its proportionality
[e]Number of symmetries between wires, $S\{C_1, C_2\}$, identified
[f]Wiring rectilinear 'area' obtained in the global routing phase
[g]Total runtime for the Wiring Planner, Symmetry Planner, Global Router phases
[h]Population size and number of generations

phases Wiring Planner, Symmetry Planner, Global Router were executed for each of the seven multiport multiterminal (MP/MT) nets, and the numerical results of the obtained port-to-port optimal rectilinear paths are presented in Table 8.9. Then, the detailed routing procedure, described in Chap. 6 of this book, automatically obtains the DRC- and LVS-correct layout. The computational times are also included in Table 8.9. Long runs were performed for the optimization-based detailed routing to improve the quality of the resulting layouts, however, feasibility, i.e., DRC and LVS-clean solutions, were obtained with less iterations.

Some areas of the automatically generated layout for design 4 of Fig. 8.13 were zoomed to detail the impact of the EM and/or WS considerations in the routing. For pairs of current/temperature that lead to an effective wire width below the minimum allowed by the fabrication process, the minimum allowed was used. For electric-current intensive nets (e.g., *vdd*, *gnd* or *out*), computing the current-correct wire width resulted, in some cases, in wires more than six times wider than the minimum allowed by the process (particularly, in the example zoomed, some sections of the

net *out* have 1345 nm). However, by using the EM-aware wire planning, the WT is optimized taken into account the correct width of the wires, leading to an efficient topology solution. Ensuring the safe current-flow in the wires is mandatory, as the circuit is compromised if the wrong wire width is used instead, which may occur when using non-EM-aware methodologies or in the manual design, especially during redesign cycles where some electric-currents may change without the corresponding update in the layout. In addition, the Wire Symmetry analysis step imposed perfect symmetries in most of the wires in the layout, which provide an additional robustness against process variations, mismatch and consequently offset errors.

Due to the inexistence of fully-automatic analog routing approaches considering both EM and WS in the state-of-the-art, a comparison with similar tools is not possible. To use the existent single-port routers in the proposed multiport test cases would require the introduction of expert knowledge in the setup process, particularly to select the most appropriated single-port among the available ports. Furthermore, the comparison with template-based, descriptive or procedural-based approaches is also impossible as these methodologies, while presenting results compliant with the designers' guidelines, are restricted to the topological relation between devices that they were intended to. However, even in those cases, they may not support wide changes in the devices' sizes.

8.3.3 Parasitic Extraction and Layout-Aware Circuit Sizing

In this section AIDA-L is used to include complete layout-related data (geometrical and parasitic) into the automatic circuit sizing, following the concepts introduced in Chap. 3 of this book. The single ended two-stage amplifier topology with a resistance in series with the capacitor is here used, as it proved to be more robust under Corner conditions (*vdd max*, temperature *min*, VCM *max*; *vdd min*, temperature *max*, VCM *max*; *vdd max*, temperature *min*, VCM *min*; and *vdd min*, temperature *max*, VCM *min*). The resistive load in this case was 10 MΩ instead of the 500 kΩ, and optimization targeted higher GBW (≥200 MHz). The objectives of the layout-aware circuit sizing optimization were again set to maximize the GDC while minimizing the real floorplan area (not estimated), simultaneously, a traditional floorplan-aware optimization-based circuit sizing was performed without considering parasitic-related data, the two resulting POFs of sizing solutions are illustrated in Fig. 8.14.

From the layout-aware POF, two designs were selected: the solution with the smallest area showing a GDC around 51 dB, and, a solution showing a GDC of around 75 dB. The solution with the largest gain was not selected because it was difficult to clearly identify either the simplified or the detailed routing over the layout. The simplified (only global routing) and detailed layouts of these two solutions were already introduced in Figs. 7.13 and 7.14 of Chap. 7 of this book. For a fair comparison, from the POF without parasitic considerations two similar designs solutions were selected: the one with the GDC closer to 51 dB (with 53.7 dB), and,

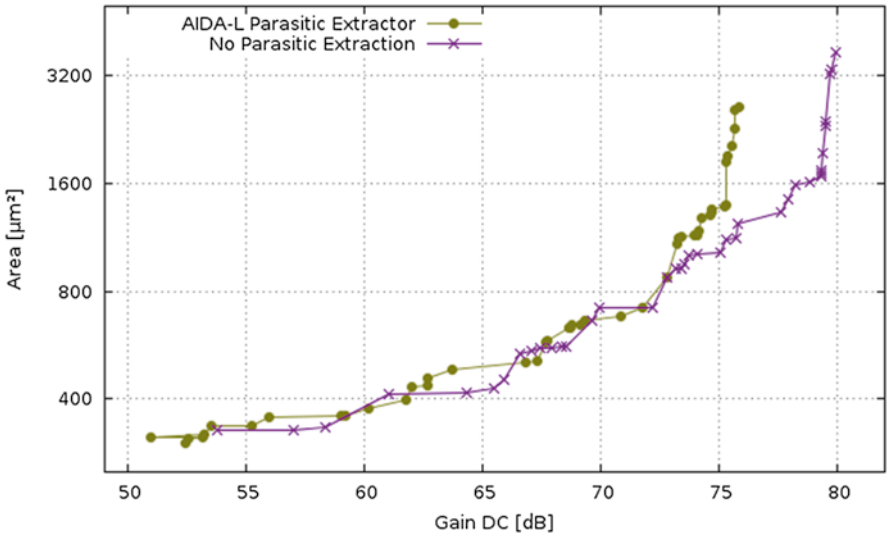


Fig. 8.14 Traditional and Layout-aware optimization POFs between the two optimization objectives: area (m^2) versus GDC (dB)

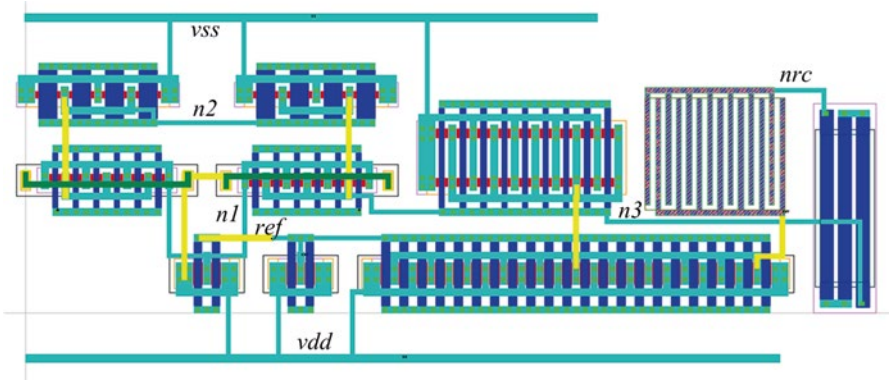


Fig. 8.15 Layout generated for the 53.7 dB design of the traditional optimization-based sizing. Layout area: 337 μm^2

another with a GDC closer to 75 dB (with 75.4 dB). The layouts for these two designs were automatically generated with AIDA-L and are presented in Figs. 8.15 and 8.16.

The performance measures before and after layout for the four different designs are summarized in Table 8.10. The impact of parasitics is not considered in the traditional optimization-based sizing, leading to post-layout designs that do not meet specifications. By considering parasitic effects during the entire optimization, what may look overdesign is in fact a feasible solution when considering the parasitic

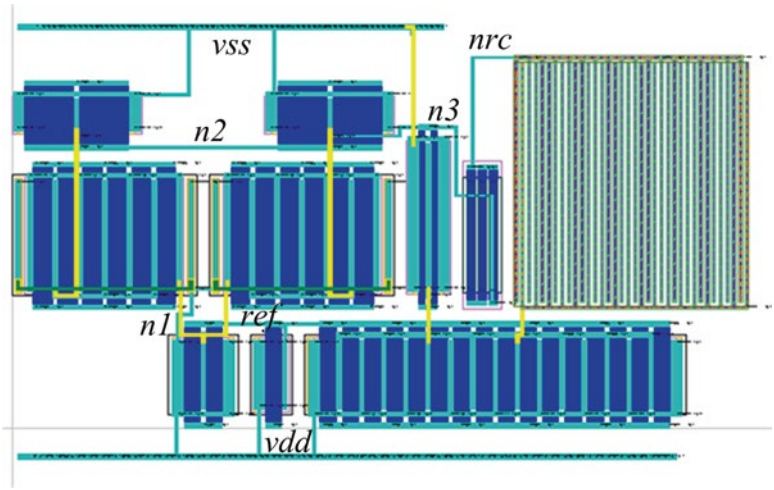


Fig. 8.16 Layout generated for the 75.4 dB design of the traditional optimization-based sizing. Layout area: 1021 μm². Not in the same scale of layout of Fig. 8.15

effects. Like in previous efforts on the area, unit-gain frequency and phase margin are the most sensitive measures for typical analog amplifiers [4].

Also, the performance comparisons with Mentor Graphics' Calibre® [5] extractions confirm the precision of AIDA-L's Parasitic Extractor, and its suitability to be included in the layout-aware circuit optimization loop, avoiding the need for in-loop detailed routing. In the first case the area of the resultant layout-aware is smaller than the one obtained by the traditional simulation-based sizing (310 μm² against 331 μm²). While in the second case, the layout-aware one has a greater area (1330 μm² against 1021 μm²) and is less compact than the layout obtained from traditional optimization-based sizing, however, this was obtained not weighting the impact of the parasitic devices in the solutions' performance.

8.4 Case Study III: Two-Stage Folded Cascode Amplifier

The third circuit example is a two-stage folded cascode amplifier of Fig. 8.17, loaded with a 10 KΩ resistor in parallel with a 1 pF capacitor and the methodologies applied to this circuit are outlined in Table 8.11.

8.4.1 Parasitic Extraction and Layout-Aware Circuit Sizing

In this section, AIDA-L is used to include complete layout-related data (geometrical and parasitic) into the automatic circuit sizing, following the concepts introduced in Chap. 3 of this book. The objectives of the layout-aware circuit sizing optimization

Table 8.10 Pre- and post-layout simulations of the single ended two-stage amplifier

		~51 dB Designs				~75 dB Designs			
		Traditional optimization-based sizing		Layout-aware sizing		Traditional optimization-based sizing		Layout-aware sizing	
Specs		Netlist	Calibre®[a]	AIDA-L's Parasitic Extractor[b]	Calibre®[c]	Netlist	Calibre®[d]	AIDA-L's Parasitic Extractor[e]	Calibre®[f]
GDC	max ≥ 50 dB	53.7	53.7	50.9	50.9	75.4	75.4	74.6	74.6
Area	min	337 μm^2		310 μm^2		1021 μm^2		1330 μm^2	
Gbw	≥200 MHz	208.69	**183.37**[g]	204.29	206.30	206.68	**182.66**[g]	209.25	210.05
Pm	≥55°	55.56	**46.42**[g]	74.27	73.54	57.83	58.27	64.51	60.56
Idd	≤200 µA	135.4	135.1	176.8	176.5	190.5	189.2	190.9	190.2
PSRR	≥55 dB	77.23	75.53	75.44	75.45	88.09	77.02	83.89	80.12
Voff	≤5 mV	3.80	3.83	4.39	4.39	4.83	4.91	5.02	4.38
No	≤600 µVrms	435.6	441.8	301.2	305.9	347.7	336.3	381.9	390.3
Sn	≤100 nV/√Hz	21.41	21.59	16.17	16.32	25.2	25.7	26.4	26.4

All layouts considered were generated using AIDA-L
[a]Extracted over the detailed layout of Fig. 8.15
[b]Extracted over the floorplan with global routing of Fig. 7.13a
[c]Extracted over the detailed layout of Fig. 7.13b
[d]Extracted over the over the detailed layout of Fig. 8.16
[e]Extracted over the floorplan with global routing of Fig. 7.14a
[f]Extracted over the detailed layout of Fig. 7.14b
[g]Constraints violated in post-layout performance

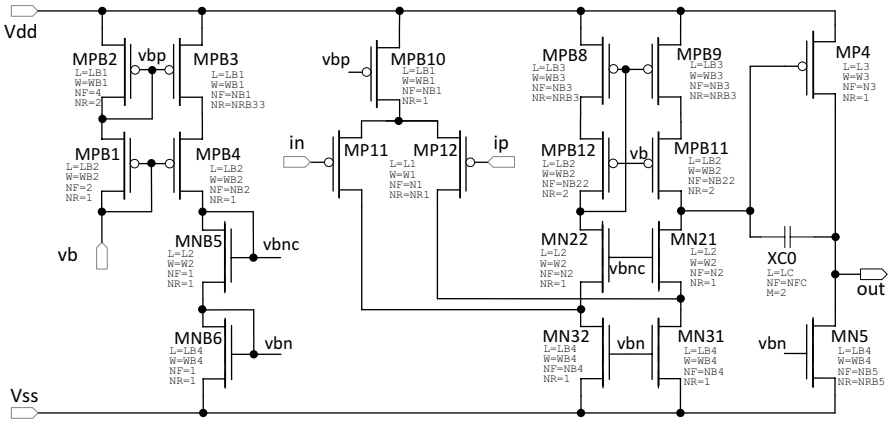


Fig. 8.17 Two-stage folded cascode amplifier schematic [16]

Table 8.11 Methodologies applied to the two-stage folded cascode amplifier

	Template-based Placer (Chap. 4)	Optimization-based Placer (Chap. 5)	Fully-automatic Router (Chap. 6)	Parasitic Extractor (Chap. 7)	Post-layout validation
Section	8.3.1	8.3.2 8.3.3	8.3.1 8.3.3	8.3.1	8.3.1 8.3.3

were set to maximize the bandwidth while minimizing the current consumption. Simultaneously, a traditional optimization-based circuit sizing was performed without considering layout-related data, and the target specifications for both optimizations are shown in Table 8.12. In Fig. 8.18 the resulting POFs of sizing solutions are illustrated. Also, the complete layout for the solutions from the traditional POF was generated, and the post-layout performances extracted. The feasible and unfeasible solutions post-layout, i.e., that meet or fail the specifications, are also highlighted in Fig. 8.18.

The performance measures before and after layout, for the solutions from both POFs with smaller current consumption are summarized in Table 8.12. By using AIDA-L in the sizing loop, the circuit is correctly sized, also, as it can be seen, performance measurements using AIDA-L's Parasitic Extractor, closely match the ones from Calibre®. In the traditional simulation-based optimization the parasitic effects are not accounted for, and for that reason, when the resulted layout is simulated with the inclusion of circuit's parasitics, the GBW specification is no longer met (whether the circuit is tested on Calibre® or on AIDA-PEx). Furthermore, from the traditional optimization-based circuit sizing, seven out of eight sizing solutions turn unfeasible in post-layout performance. In Fig. 8.19, the resulted layouts for the sizing used in Table 8.12 are illustrated. Although the layouts are alike, the traditional one doesn't fulfill the specifications as the layout-aware one does, due to the inclusion of physical effects in the circuit sizing phase.

Table 8.12 Pre- and post-layout simulations of the two-stage folded cascade amplifier

Specs		Traditional optimization-based sizing			Layout-aware sizing	
		Netlist	AIDA-L's Parasitic Extractor[a]	Calibre®[b]	AIDA-L's Parasitic Extractor[a]	Calibre®[c]
Idd	min (mA)	5.35	5.35	5.35	5.62	5.62
Gbw	≥ 400 MHz	402.07	**393.92**[d]	**394.48**[d]	407.12	409.17
GDC	≥70 dB	73.53	73.53	73.53	75.68	75.74
Area	≤20k μm^2	12.09k	17.07k		16.96k	
No	≤500 μVrms	197.05	195.81	194.84	196.09	194.93
Pm	≥55°	57.97	55.55	55.31	56.36	56.88
Ov[e]	≥50 mV	59.48	59.48	59.48	50.61	50.48
D[f]	≥100 mV	100.33	105.06	105.06	105.37	105.06

All layouts considered were generated using AIDA-L
[a]Extracted over global routing solutions
[b]Extracted over the detailed layout of Fig. 8.19a
[c]Extracted over the detailed layout of Fig. 8.19b
[d]Constraints violated in post-layout performance
[e]Overdrive voltage
[f]Saturation voltage

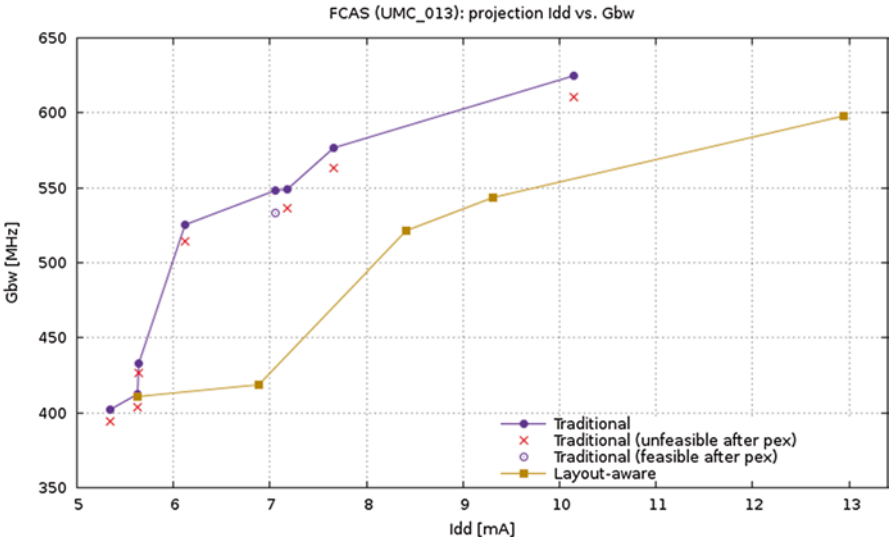


Fig. 8.18 Traditional and layout-aware optimization POFs [16]

The computational time of each task implemented in AIDA-L for the layout of Fig. 8.19b is presented in Table 8.13. When comparing to previous case studies, this circuit presents a greater challenge for the layout generation with AIDA-L, still, the execution time (without detailed routing) are perfectly sustainable in-loop.

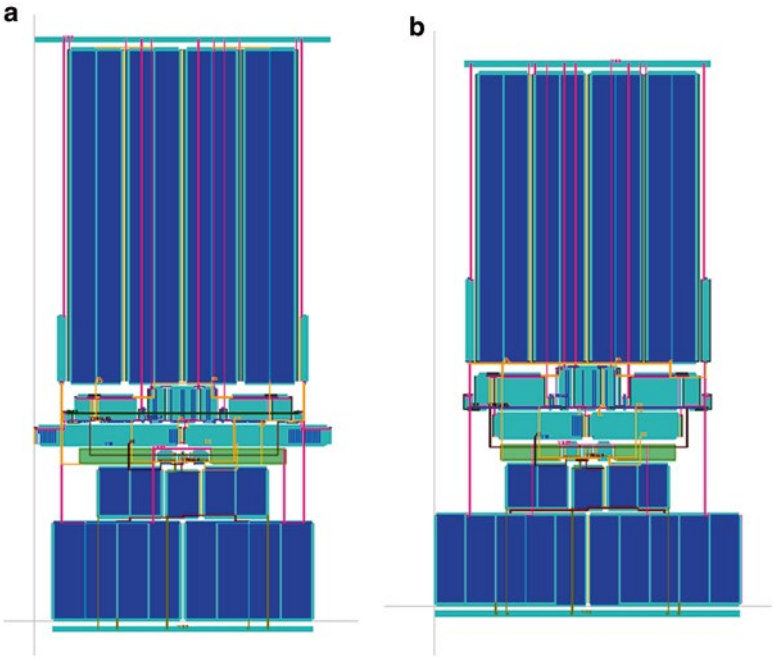


Fig. 8.19 Layouts generated for the: (**a**) Traditional optimization-based circuit sizing, layout area: 17.07 k μm^2. (**b**) Layout-aware optimization, layout area: 16.96 k μm^2 [16]

Table 8.13 Execution time for each task implemented in AIDA-L for the two-stage folded cascade amplifier

	AIDA-L			
	Template-based Placer (s)	Global routing[a] (s)	Detailed routing (s)	Parasitic Extractor (s)
Fig. 8.18b	~0.02	~0.04	~540	~2

Computational times obtained on a Fedora Virtual Machine running on Intel® Core™ i7-3770 @ 3.4 GHz with 32 GB of RAM

[a]Global routing times encompasses: Wiring planner, Symmetry planner and Global routing

8.4.2 Layout Generation: Optimization-Based Placer

In previous section, AIDA-L was used to include layout-related data into the automatic circuit sizing. The circuit sizing was optimized, with AIDA-L template-based Placer and several template files, resulting in the solutions described before. In this section the floorplan of the layout-aware solution detailed in Table 8.12 is (re)generated using the optimization-based Placer. The design hierarchy was

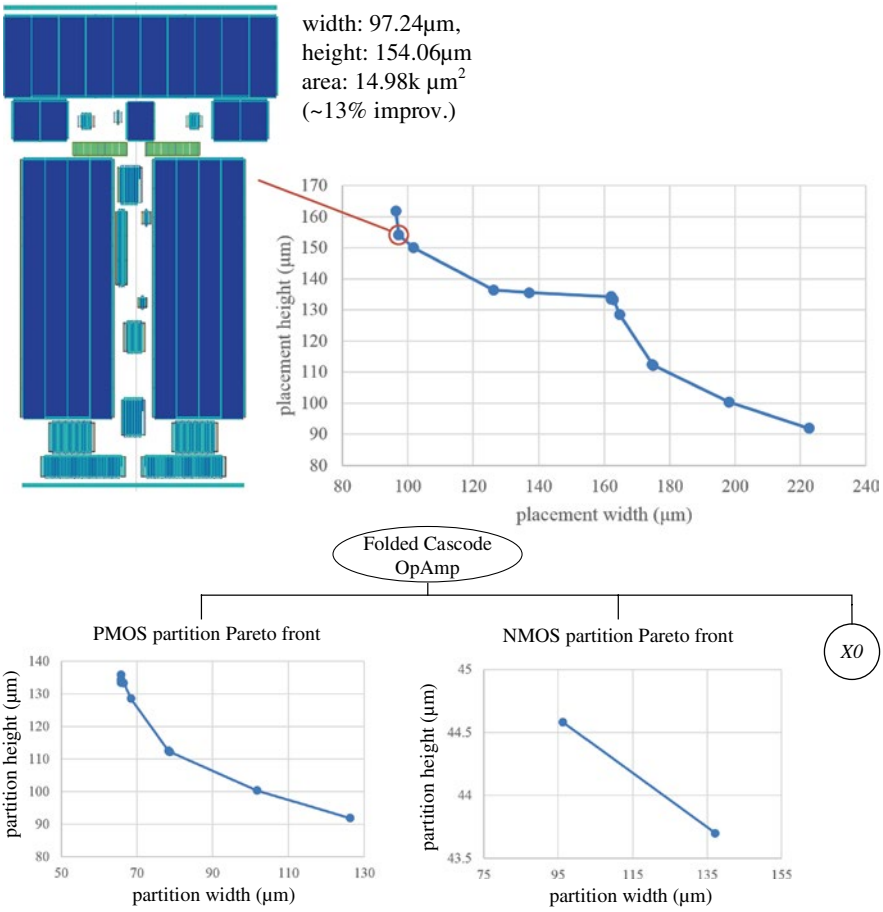


Fig. 8.20 Design hierarchy of the folded cascade amplifier: POFs with the tradeoff placement height versus width for the PMOS/NMOS partition are combined at the level, which presents 14 different placement solutions with area improvements of the original design [15]

defined with two partitions containing the PMOS and NMOS devices separately. The symmetry pairs of the original design were kept, and, at the top partition the resultant Pareto fronts of the sub-partitions are combined along with the MOM capacitor. The resulting fronts for the complete design hierarchy are illustrated in Fig. 8.20.

They were generated in 9 seconds on an Intel® Core™ i7-3770 @ 3.4 GHz with 32 GB of RAM. The top front provides 14 different placement solutions for the amplifier, representing the different aspect ratios found from the exploration of the tradeoff placement's *width* versus *height*. Some of the solutions improved the circuit area of the previous design, up to an improvement of 13 %.

Table 8.14 Current-flow constraints for the two-stage folded cascode amplifier, i.e., paths from power to ground lines

Current-flow constraints
$V_{dd} \rightarrow \text{MPB}_3 \rightarrow \text{MPB}_4 \rightarrow \text{MNB}_5 \rightarrow \text{MNB}_6 \rightarrow V_{ss}$
$V_{dd} \rightarrow \text{MPB}_{10} \rightarrow \text{MP}_{11} \rightarrow \text{MN}_{32} \rightarrow V_{ss}$
$V_{dd} \rightarrow \text{MPB}_{10} \rightarrow \text{MP}_{12} \rightarrow \text{MN}_{31} \rightarrow V_{ss}$
$V_{dd} \rightarrow \text{MPB}_8 \rightarrow \text{MPB}_{12} \rightarrow \text{MN}_{22} \rightarrow \text{MN}_{32} \rightarrow V_{ss}$
$V_{dd} \rightarrow \text{MPB}_9 \rightarrow \text{MPB}_{11} \rightarrow \text{MN}_{21} \rightarrow \text{MN}_{31} \rightarrow V_{ss}$
$V_{dd} \rightarrow \text{MP}_4 \rightarrow \text{MN}_5 \rightarrow V_{ss}$

8.4.3 Layout Generation: Optimization-Based Placer with Current-Flow and Current-Density Considerations

While previous section shows the capabilities of the optimization-based Placer to compact and present multiple floorplans for a larger number of devices, without designer's guidelines, it is unlikely that an automatically generated floorplan without considerations besides placement area can match the performance of an expert made design, as overviewed in Case Study I. In the next optimization, the placement is constrained with the current-paths presented in Table 8.14, i.e., paths from power to ground lines, following the principles introduced in Chap. 5 of this book.

The optimization of the top partition resulted in a front with 50 non-dominated placement solutions, all verifying the current-flow constraints, from the exploration of the tradeoff placement's *width*, placement's *height* and sum of the optimal EM-aware *WTs*. The projections for each pair of objectives are presented in Fig. 8.21, and, the floorplan solution the lowest of *WT* is also highlighted. The placement solutions of the top front of Fig. 8.20 are plotted against the placement's *width* versus placement's *height* projection. As expected, some areas of the Pareto front without current considerations are inaccessible due to the inclusion of current-flow constraints (i.e., solutions with higher values of placement's *width*).

The selected floorplan was routed and the post-layout simulation results are found in Table 8.15. The post-layout simulation of the layout obtained using the template-based placer is shown again for an easy side-by-side comparison. The layout resultant from the optimization-based placement outperforms the template-based in terms of GBW, and, with a 12 % improvement on the layout area.

The relevant computational times are presented on Table 8.16. The increase in the execution time of the optimization-based Placer with current-flow and current-density considerations when compared to the template-based, and also, optimization without current considerations, is rewarded with a Pareto front with more and better placement solutions. While this execution time is impossible to be included in the layout-aware loop, these results reinforce the usage of the optimization-based Placer for the standalone aspect of AIDA-L, as to, post-optimize the resulting layouts from the parasitic-aware fronts.

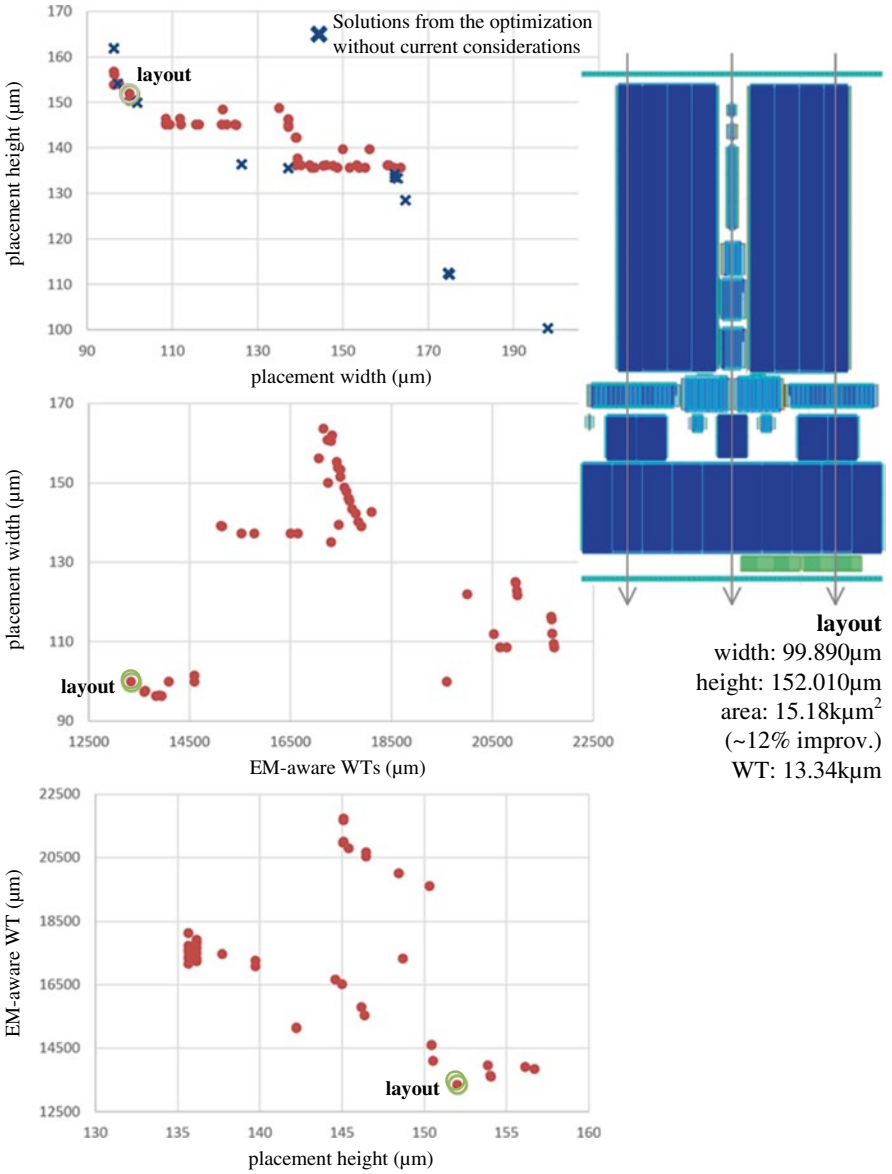


Fig. 8.21 Placement optimization of the two-stage folded cascade amplifier with current-flow constraints for the tradeoff placement's width, height and sum of the optimal EM-aware WTs. Different projections of the 50 different placement solutions found [15]

Table 8.15 Post-layout performances for two-stage folded cascade amplifier

Performance	Spec.	Post-layout[a]	
		Optimization-based floorplan	Template-based floorplan
Idd	min (mA)	5.62	5.62
Gbw	≥400 MHz	410.40	409.17
GDC	≥70 dB	75.74	75.74
Area	≤20k μm^2	15.18k	16.96k
No	≤500 μVrms	196.12	194.93
Pm	≥55°	56.81	56.88
Ov[b]	≥50 mV	50.18	50.48
D[c]	≥100 mV	109.32	105.06

[a]DRC, LVS and extraction performed in Mentor Graphics' Calibre®
[b]Overdrive voltage
[c]Saturation voltage

Table 8.16 Execution time for each task implemented in AIDA-L for the two-stage folded cascade amplifier

	AIDA-L		
	Placer (s)	Global routing (s)	Detailed routing (s)
Template-based	~0.02	~0.04	~540
Optimization-based without current considerations	~9		
Optimization-based with current-flow and current-density	~400		

Computational times obtained on an Intel® Core™ i7-3770 @ 3.4 GHz with 32 GB of RAM

8.5 Case Study IV: Operational Transconductance Amplifier

The last circuit example is an OTA used by Edinei et al. in a system with two synchronized phase-locked loops for built-in self-testing of high speed analog-to-digital converters [6]. The schematic is presented in Fig. 8.22 and is composed by 40 devices in the main amplifier and bias circuit, and each of the common mode feedback sub-circuits contains 46 devices. In total, the circuit's floorplan was generated by AIDA-L's template-based Placer considering the 112 devices, using the devices' sizes and relative positioning (Fig. 8.23) of the original design and handmade layout. The generation of the complete layout using the proposed methodologies was performed, as in the original design, for the UMC 130 nm CMOS design process.

Again, the EM-aware wire planning, multilayer multiport selection, Steiner point assignment and detailed routing steps were executed for each of the 41 MP/MT nets, each one connecting from a range of 2–22 multiport terminals, and the resulting layout is presented in Fig. 8.24. Furthermore, 15 of the 41 nets were selected and

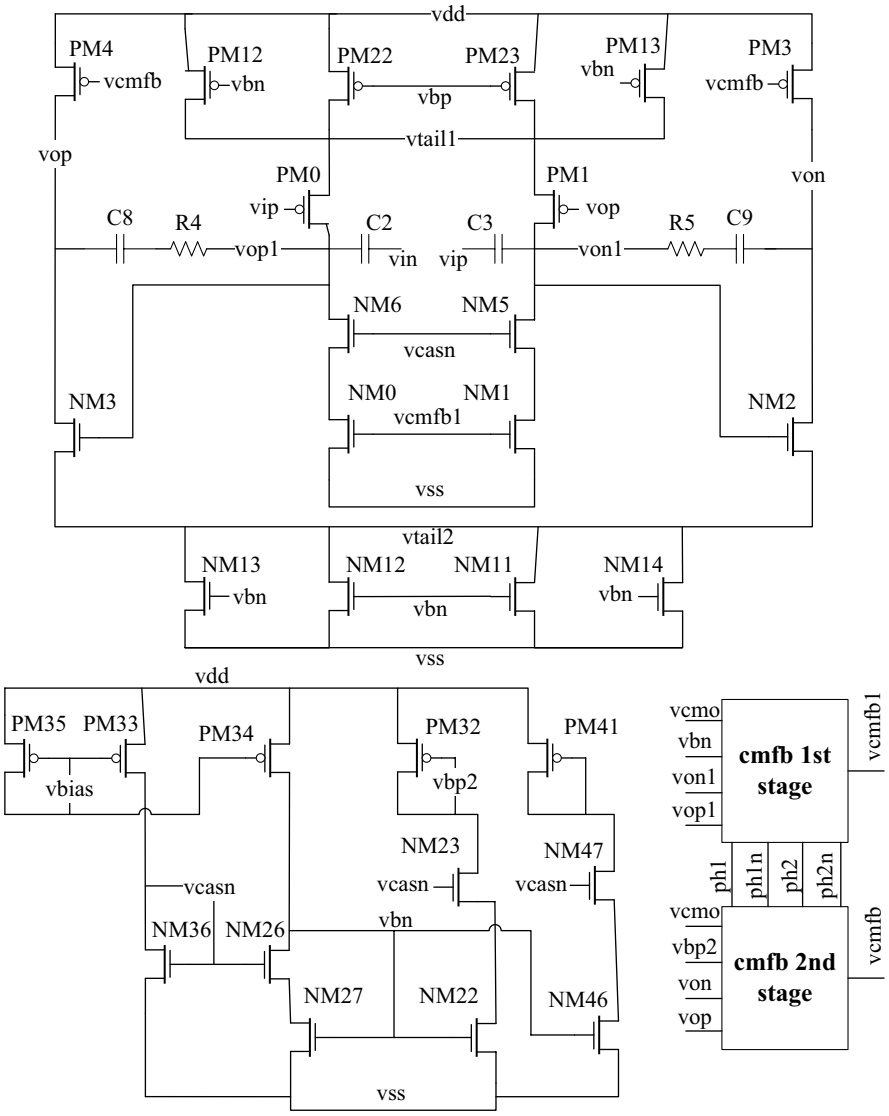


Fig. 8.22 OTA schematic

the numerical results of the respective port-to-port optimal rectilinear forests are presented in Table 8.17. The single-net and multi-net procedures for detailed routing took approximately 5 hours to complete.

Even though the huge dimension difference of the circuit when compared to the three circuits previously presented, it is possible to observe that the number of terminals for each net was kept relatively small, and did not increased proportionally. This was due to the use of transistors' stacks, merged structures and interdigitized

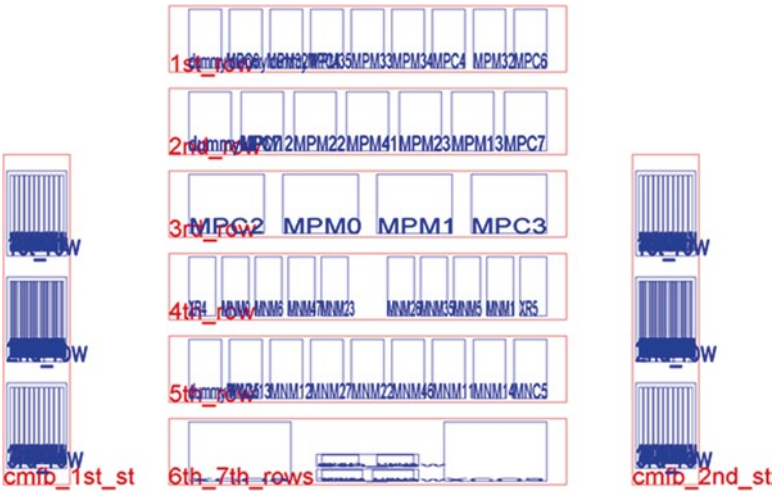


Fig. 8.23 Topological relations between cells of the hierarchical template

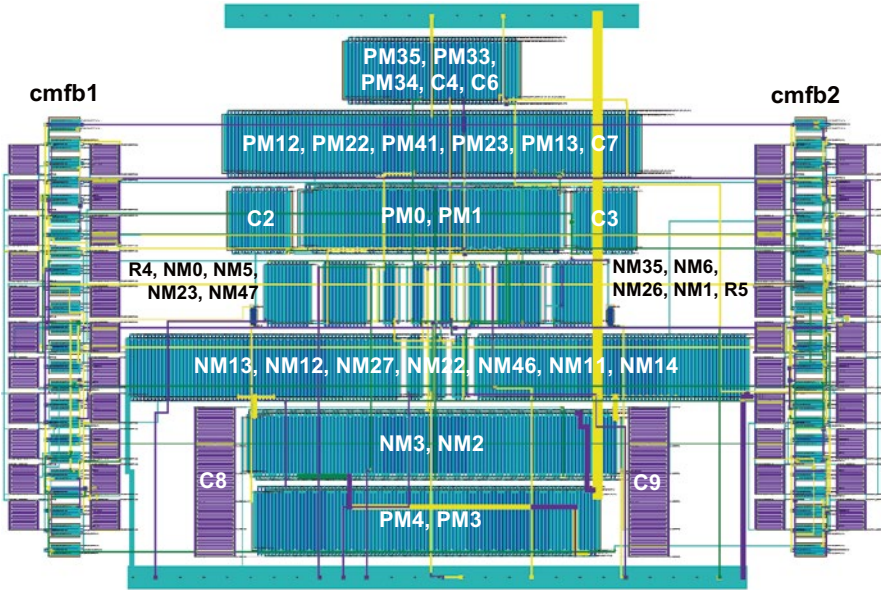


Fig. 8.24 Automatically generated EM-aware routing for the OTA [17]

layout styles from AIDA-AMG, on which some common terminals are connected/overlapped at the generation of the modules, and not during the routing process. These layout styles do not only save space, but they also improve the circuit performance by decreasing parasitic effects, and as shown, end up decreasing the total number of terminals that needs to be routed. Steiner points are also crucial to obtain

Table 8.17 Results obtained by using the EM-Aware MP/MT routing approach in the operational transconductance amplifier

Net	#Terminals	#Ports[a]	#Layers/V_{cost}[b]	MP EA-WP area[c] (nm.A)	#Connectivity[d]	Runtime (s)	MP-selection area[e] (nm.A)	Runtime (s)	Steiner points	Runtime (s)
vip	3	[9–19]	6/5	67,473	2	<0.01	10,152	0.01	0	<0.01
vop1	10	[1–35]	6/5	5,228,124	9	0.01	2,503,710	0.02	4	<0.01
von1	10	[1–35]	6/5	7,429,136	9	<0.01	4,672,646	0.02	4	<0.01
vcasn	8	[3–10]	6/5	1,974,208	9	0.01	1,626,065	0.01	5	<0.01
vcmo	16	[1–4]	6/5	150,286	15	0.02	147,007	0.01	9	<0.01
vbp2	7	[1–6]	6/5	2,636,981	6	0.01	2,416,992	0.01	5	<0.01
vbn	11	[1–42]	6/5	1,990,155	10	0.01	1,554,786	0.01	6	<0.01
vbp	3	[3–60]	6/5	1,486,545	2	<0.01	1,108,641	0.01	1	<0.01
vcmfb	7	[1–50]	6/5	499,538	6	<0.01	328,128	0.02	5	<0.01
vcmfb1	8	[1–10]	6/5	82,573	7	<0.01	74,429	0.01	3	<0.01
vtail1	5	[5–12]	6/5	$10e^6$	4	<0.01	2,022,603	0.01	2	<0.01
vop	8	[1–51]	6/5	$17e^6$	7	<0.01	$16e^6$	0.02	3	<0.01
von	8	[1–51]	6/5	$18e^6$	7	<0.01	$17e^6$	0.02	4	<0.01
vdd	7	[1–61]	6/5	$233e^6$	6	<0.01	$215e^6$	0.01	2	<0.01
vss	22	[2–41]	6/5	$198e^6$	22	0.02	$105e^6$	0.05	5	<0.01

Computational times obtained on an Intel® Core™ i7-3610 @ 2.3 GHz with 8 GB of RAM

[a]Range of number of ports for each terminal

[b]Cost of assigning a via (for the enhanced A* search)

[c]Wiring '*area*' of the EM-aware WT, computed by Eq. (6.6) on Chap. 6 of this book. Where l_i is the length (in nanometers) and j_i the current density (in Amperes) of the wire i. In the traditional EM-aware problem the current j_i is used instead of the wire's width due to its proportionality

[d]Number of terminal-to-terminal connections in the strongly connected network

[e]Wiring rectilinear 'area' obtained in the global routing phase

less congested layouts, where dozens of terminal-to-terminal connections are removed from the strongly connected networks. By removing two connections of the same net in favor of a single larger connection, with the sum of current-flows, the task of the detailed routing step is alleviated twofold by halving the number of possible design rule errors and short circuit violations, and decreasing the number of wires/shapes involved in the verification.

8.6 Benchmarks

As stated, to further study the behavior of the proposed techniques, in addition to the presented circuit designs, three benchmarks were considered, two for placement and one for routing.

8.6.1 Optimization-Based Placer Benchmark: Single-Objective vs. Multi-Objective

To benchmark traditional single-objective (SO) absolute approaches overviewed in the state-of-the-art, in Chap. 2 of this book, versus multi-objective (MO) absolute placers, including the optimization-based Placer proposed in Chap. 5 of this book, a case study containing 15 cells according to Table 8.18 was adopted, where δ is the multiplier for width and height values of each cell. Note that solutions without wasted area are not possible for this problem.

Table 8.18 Generic case study with 15 cells

Name	Type	Width	Height
A1	Autonomous c^a	δ	δ
A2		2δ	δ
A3		2δ	2δ
A4		3δ	δ
A5		3δ	2δ
AS1	Self-symmetric c^s	δ	δ
AS2		3δ	2δ
AS3		2δ	3δ
AS4		4δ	δ
S1	Symmetry Pair (c^1, c^2)	δ	3δ
S2		2δ	3δ
S3		2δ	δ
● *cell* c_i *area*		$55\delta^2$	

Table 8.19 Comparison in terms of placement area, $f_2(x)$, of the generic case study (with $\delta = 100$) between SO algorithms (SHC, SA and GA) and MO algorithms (proposed CAMOSA and NSGA-II), for 100 runs each

		#Runs that converged to a solution with $f_2(x)$				Best placement area found in all runs
		<610,000	<630,000	<670,000	<700,000	
SO[a]	SHC[b]	2	21	78	99	600,400
	SA[c]	8	9	53	93	595,000
	GA[d]	0	1	3	13	612,864
MO[e]	CAMOSA[f]	11	43	98	100	576,802
	NSGA-II[g]	1	6	26	56	604,314

[a]Single-objective algorithms were set to minimize the cost function: min $f_2(x) + w * g_0(x)$, i.e., minimization of placement area with weighted illegal overlaps. w was set to 100, experimentally chosen at priori

[b]he probability of acceptance of a new solution for the SHC is given by Eq. (2.2). T was set to 0.01, which is the best temperature value found empirically after a complete T sweep for the current problem. The number of evaluations for the SHC is the number of iterations

[c]The probability of acceptance of a new solution for the SA is given by Eqs. (2.2) and (2.3). The parameters were set to $T_{MAX} = 10$, $R = 0.01$ and $k = 500$, which are the best values found empirically after a complete parameter sweep for the current problem. The number of evaluations for the SA is the number of iterations

[d]The parameters of the GA were set to: population size = 64, crossover rate = 90 % and mutation rate = 3 %, found empirically after a complete parameter sweep for the current problem. The number of evaluations for the GA is (*generations*population_size*)

[e]Multi-objective algorithms were set to minimize both placement's width, $f_0(x)$, and placement's height, $f_1(x)$, constrained to $g_0(x) = 0$

[f]The parameters the CAMOSA were set to: $T_{FMAX} = 1$, $T_{GMAX} = 0.01$, $R = 0.1$ and $k = 1000$ are the best values found empirically after a complete parameter sweep for the current problem. The number of evaluations for each run is the number of iterations

[g]The parameters of the NSGA-II were set to: population size = 64, crossover rate = 90 % and mutation rate = 5 %, found empirically after a complete parameter sweep for the current problem. The number of evaluations for the NSGA-II is (*generations*population_size*)

In order to study the convergence of the different algorithms applied to placement optimization, three classical SO algorithms, namely, stochastic hill climber (SHC), simulated annealing (SA) and genetic algorithm (GA) were tested in the provided case study and the results are compared with the MO algorithms, the proposed CAMOSA of Chap. 5 of this book and NSGA-II [7]. NSGA-II was selected to represent the class of population-based evolutionary algorithms, it was selected over SPEA and other MO evolutionary algorithms due to the good characteristics of the output Pareto front. For a proper comparison, all the results are presented for the value of placement area, i.e., $f_2(x)$. Results are presented in Table 8.19 and each line corresponds to 100 executions of the algorithm under test for the same number of evaluations.

After the conclusion of each optimization process, a linear-time packing deterministic procedure is performed to compact the placement in x, followed by another procedure to compact the placement in y. This simple procedure does not improve significantly the area usage, but eliminates any white spaces badly explored between

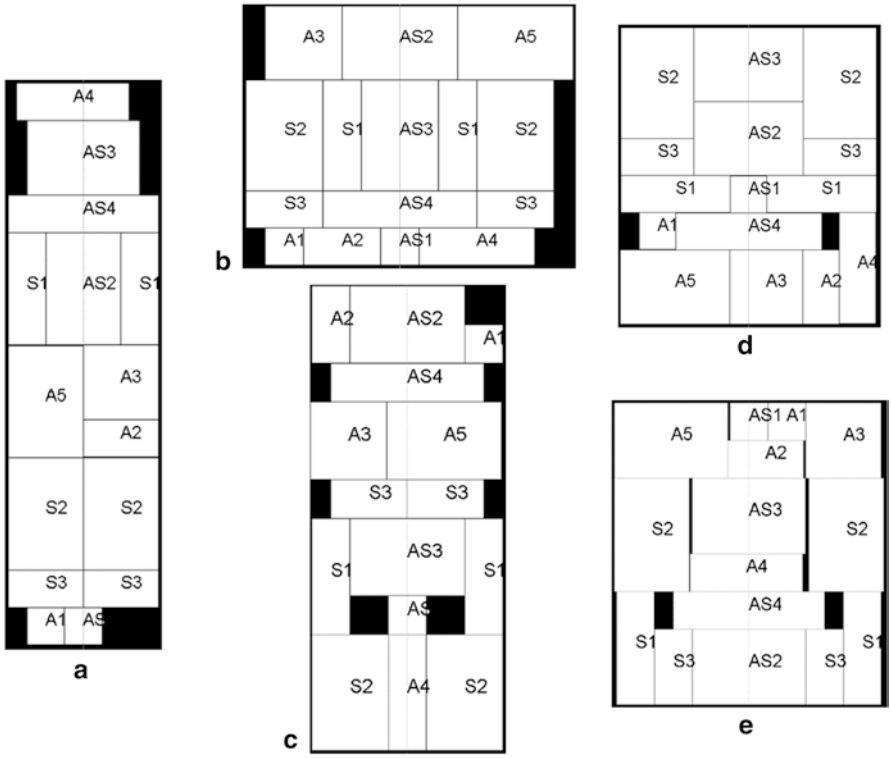


Fig. 8.25 Illustration of the best solutions found in 100 runs for each algorithm: (**a**) SHC, $f_2(x) = 600{,}400$; (**b**) SA, $f_2(x) = 595{,}000$; (**c**) GA, $f_2(x) = 612{,}864$; (**d**) CAMOSA, $f_2(x) = 576{,}802$; and (**e**) NSGA-II, $f_2(x) = 604{,}314$ [18]

cells. In Fig. 8.25 are illustrated the best placement solutions found for each algorithm in all the 100 runs.

As expected, hill climber-based algorithms, i.e., SHC, SA and CAMOSA, perform extremely better than population-based algorithms for analog placement optimization problem. CAMOSA outperforms all the other single- and multi-objective algorithms for the same number of runs and iterations in terms of placement area. Furthermore, due to the MO nature, the results of each run is a Pareto front of non-dominated placement solutions, representing the tradeoff between placements' width, $f_0(x)$, and placements' height, $f_1(x)$. The proposed concept has also the advantage of do not requiring an error-prone weighted cost function (with objectives and constraints normalized) to be minimized. In Fig. 8.26 are presented the 100 different Pareto fronts, obtained with CAMOSA and NSGA-II. Note that obviously some fronts are dominated by others, however, it is useful to illustrate the surface of the solution space explored by the algorithm in different executions with non-fixed seed. CAMOSA presents a better spread of solutions and better convergence near the true Pareto-optimal front.

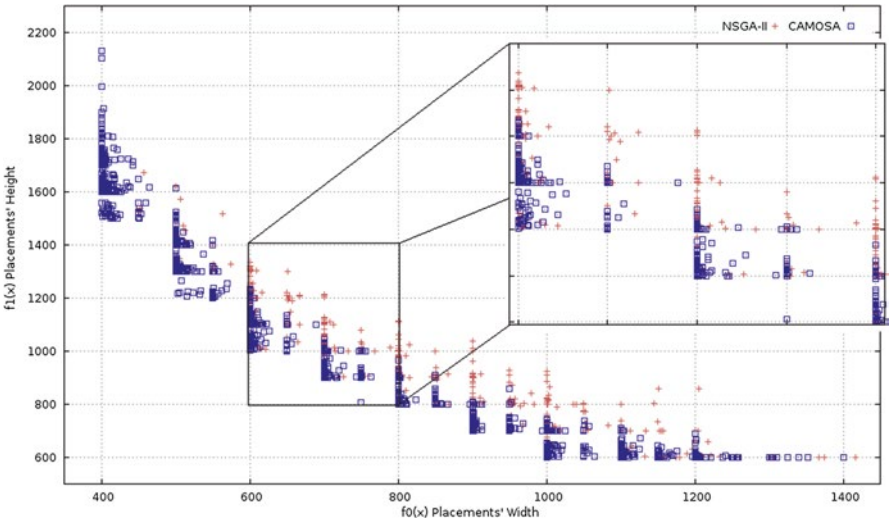


Fig. 8.26 Pareto fronts of non-dominated placement solutions obtained with 100 random runs with non-fixed seed with CAMOSA and NSGA-II [18]

Table 8.20 MCNC benchmark circuits' description

Circuit	#Mod.	#Sym. mod.	Mod. area (mm^2)	Description
apte	9	8 (4 sym. pairs)	46.56	Eight large blocks and one small
hp	11	8 (4 sym. pairs)	8.83	Large variations in block sizes and aspect ratios
ami33	33	6 (3 sym. pairs)	1.16	
ami49	49	4 (2 sym. pairs)	35.45	Uniform distribution of block sizes and aspect ratios

8.6.2 MCNC Benchmarks: Optimization-Based Placer vs. Topological

Four MCNC benchmarks, i.e., Microelectronics Center of North Carolina (MCNC) benchmarks, for analog placement design (*apte*, *hp*, *ami33* and *ami49*) are used to compare the optimization-based Placer proposed in Chap. 5 of this book with the most recent topological representations, overviewed in Chap. 2 of this book. The MCNC benchmarks are described on Table 8.20, the number of modules ("#Mod."), the number of modules with symmetry requirements ("#Sym. Mod.") included in a proximity group and the total area of the modules ("Mod. Area"). Table 8.21 presents the placement area ("Area") and computational time ("Time") obtained with the proposed approach and the most recent topological representations: Sequence pair [8], Segment tree [9], TCG-S [10], Sequence-pair with dummy nodes [11] and HB*-tree [12]. The results of the works are all taken from [12]. Note that previous works pushed to the limits the optimization of these circuits, however, the proposed approach presents very competitive results in all benchmarks and with the enhancement of

Table 8.21 MCNC benchmark circuits': comparison of placement area and computational time results of the MOO (tradeoff placement's width versus placement's height) of the proposed absolute representation approach, and, the SO optimization of several topological representations: sequence-pair, segment tree, TCG-S, sequence-pair with dummy nodes and HB*-tree

	Single-objective optimization[a]										Multi-objective optimization[b]		
	Sequence pair [8]		Segment tree [9]		TCG-S [10]		SP w/dummy nodes [11]		HB*-tree [12]		AIDA-L's optimization-based Placer		
MCNC circuit	Area (mm^2)	Time[c] (s)	Area (mm^2)	Time[c] (s)	Area (mm^2)	Time[c] (s)	Area (mm^2)	Time[d] (s)	Area (mm^2)	Time[d] (s)	#Pareto front	Best area[e] (mm^2)	Time[f] (s)
apte	48.12	25	47.52	11	47.52	3	46.92	13	46.92	2	15	46.92	<1
Comparison	1.03	–	1.01	–	1.01	–	1.00	–	1.00	–	–	1.00	–
hp	9.84	138	9.71	62	9.71	50	9.43	13	9.35	2	31	9.35	3
Comparison	1.05	–	1.04	–	1.04	–	1.01	–	1.00	–	–	1.00	–
ami33[g]	1.24	684	1.23	307	1.21	423	1.24	23	1.23	12	18	1.21	19
Comparison	1.03	–	1.02	–	1.00	–	1.03	–	1.02	–	–	1.00	–
ami49[g]	37.82	2038	37.31	983	37.04	1247	38.32	29	36.85	20	14	38.17	44
Comparison	0.99	–	0.98	–	0.97	–	1.00	–	0.97	–	–	1.00	–

[a]The output of each run is a single placement solution resultant from the minimization of placement area

[b]The output of each run is Pareto front with *n* ("#Pareto front") different placement solutions, representing the tradeoff between placement's width and placement's height minimization

[c]Computational time obtained on Sun SPARC Ultra 60 @ 433 MHz

[d]Computational time obtained on Pentium™ 4 @ 3.2 GHz

[e]The solution selected from the front is the one with the smallest placement area, i.e., *width*height*

[f]Computational time obtained on Intel® Core™ i7 @ 3.4 GHz

[g]Due to the large unfeasible space a simple hill-climbing technique is used to initialize the archive, accepting one solution only if it dominates the previous one, as suggested in AMOSA [13]

providing a full set of optimal solutions to the designer, due to its multi-objective optimization (MOO) nature, instead of only one placement solution.

In order to illustrate the output provided by the proposed approach in the MCNC benchmarks, the obtained Pareto fronts of non-dominated placement solutions are illustrated in Fig. 8.27. For the *apte* benchmark, each of the 15 non-dominated solutions can be depicted in Fig. 8.28, while only the best solutions in terms of placement area for the *hp*, *ami33* and *ami49* benchmarks are presented in Fig. 8.29. For the *apte*, *hp* and *ami33* the optimal solutions in terms of placement area (46.92 mm^2, 9.35 mm^2 and 1.21 mm^2, respectively) were found and are highlighted on the fronts. The approach matches the results of the best topological representations published, and furthermore, providing several placement alternatives ("#Pareto front") for the designer to choose from.

8.6.3 Routing Benchmark: Single-Port vs. Multiport

To benchmark the MP routing methodology proposed in Chap. 6, a set of eight analog routing test cases were adopted, where one is the example introduced in Chap. 6 of this book and the other seven are randomly generated examples that are detailed in Table 8.22. The number of terminals ranges from 7 to 100 and were placed on a floorplan with variable area. In the test cases including MP terminals, the number of ports ranges from 2 to 16. Balanced positive and negative electric-current values were injected in each of the terminals, in order to satisfy the Kirchhoff's current laws. The number of obstacles ranges from 4 to 75 and were randomly distributed by the different fabrication layers, which were set to 4, also, the cost of assigning a via in the global routing process (introduced in Chap. 6 of this book) was set to 5.

Since in the proposed approach a terminal geometry has any number of ports in random dispositions, and also, on different fabrication layers, the approach taken was to transform each MP structure in a single-port structure, by randomly fixing one of the ports, and block the access to the remaining. The implementation was tested for a random net with 100 multiport terminals to prove the scalability of the solution; however it is unlikely to have nets in analog cells connecting that amount of terminals.

As it can be seen by the similarities in the strongly connected wiring topologies obtained for the single port and MP runs, EM-aware wire planning is relatively independent of the MP problem, validating the use of the geometrical center of the terminal's ports in the MP case. However, by taking advantage of the MP an average reduction of 25 % in the rectilinear wire area of the seven random examples is achieved, without a significantly increase in the execution time, given the efficiency of the pathfinding algorithm implemented.

8.7 Conclusion

Several test cases were addressed to validate the implemented methodologies. A single stage amplifier with gain enhancement using voltage combiner, a single-ended two-stage OpAmp, and a two-stage folded cascade amplifier were used to

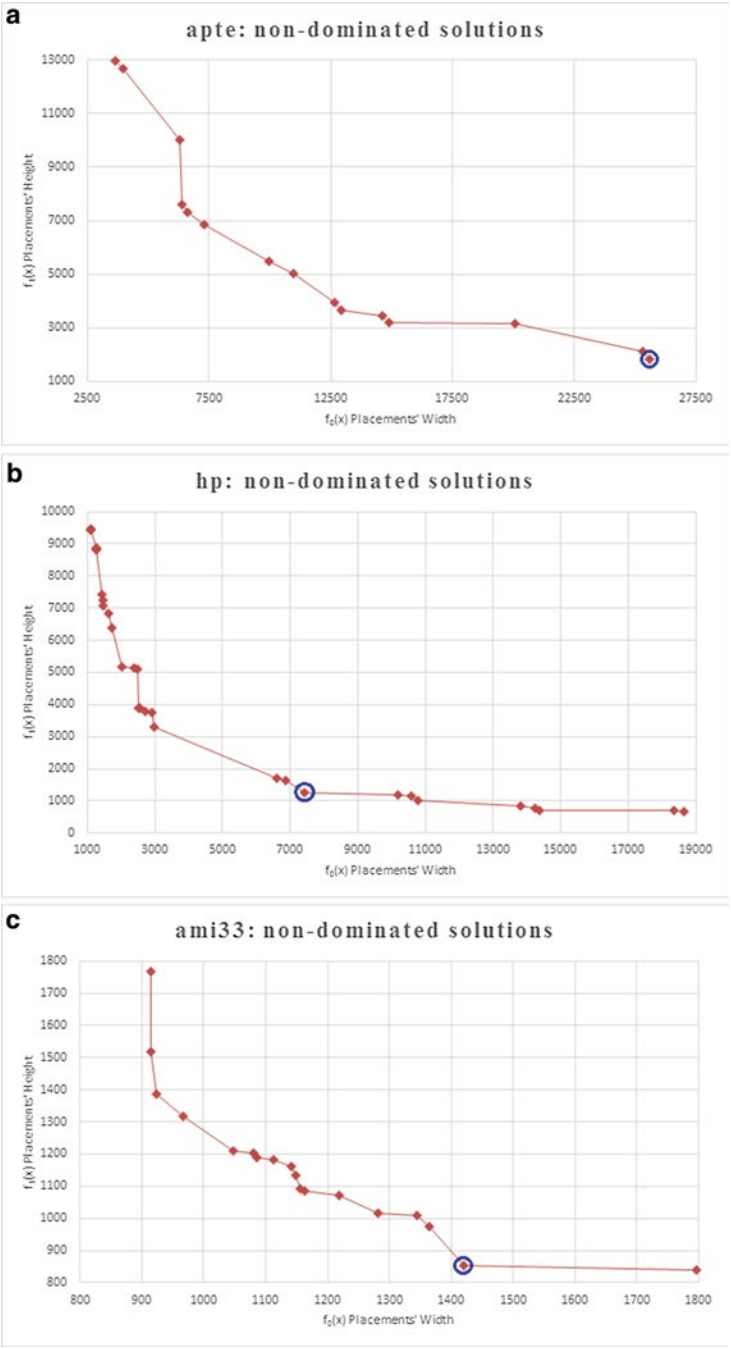


Fig. 8.27 Non-dominated placement solutions of the Pareto front placement's width versus height: (**a**) *apte* benchmark, 15 different solutions; and (**b**) *hp* benchmark, 31 different solutions. (**c**) *Ami33* benchmark, 18 solutions; and (**d**) *ami49* benchmark, 14 solutions. Solutions that minimize the placement area, $f_2(x)$, are highlighted [18]

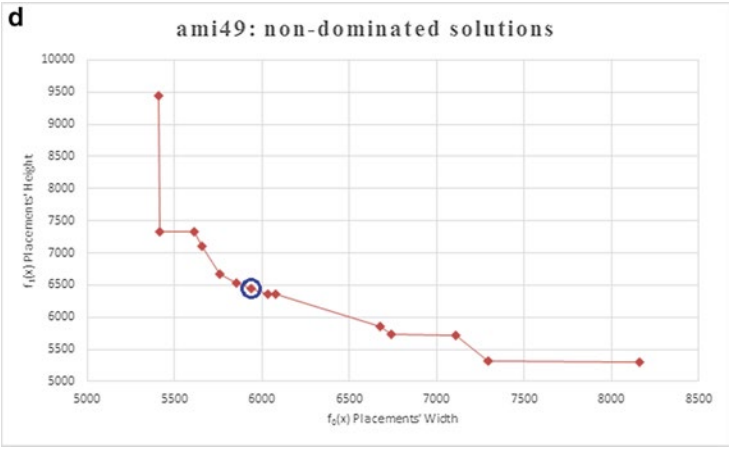


Fig. 8.27 (continued)

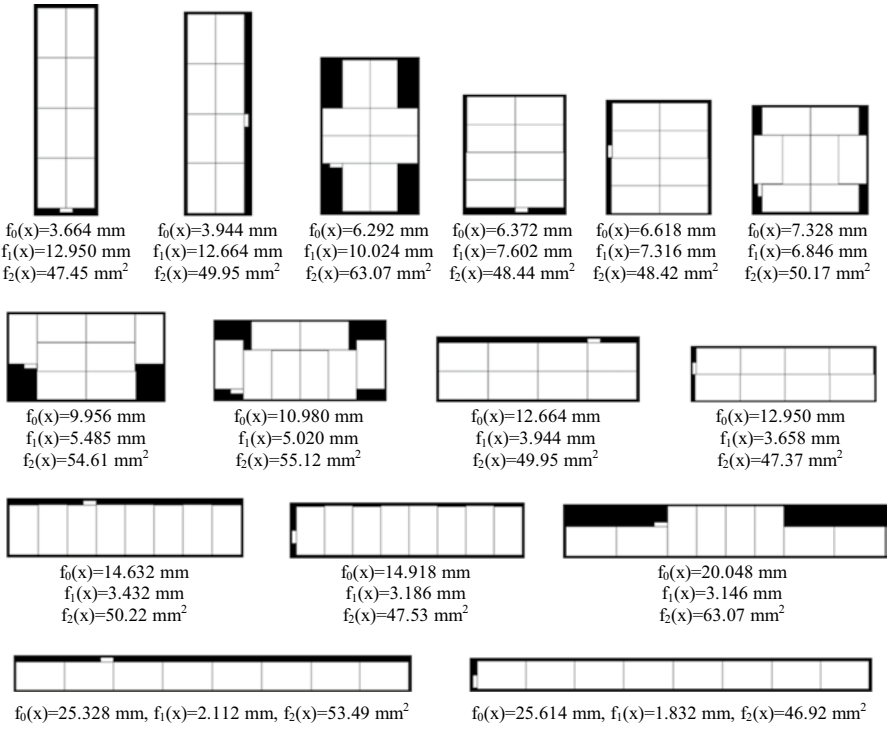


Fig. 8.28 *Apte* benchmark: illustration of the 15 non-dominated placement solutions in the obtained Pareto front of Fig. 8.27a. The solution $f_2(x) = 46.92$ mm^2 is the one that minimizes the placement area. All placed at the same scale [18]

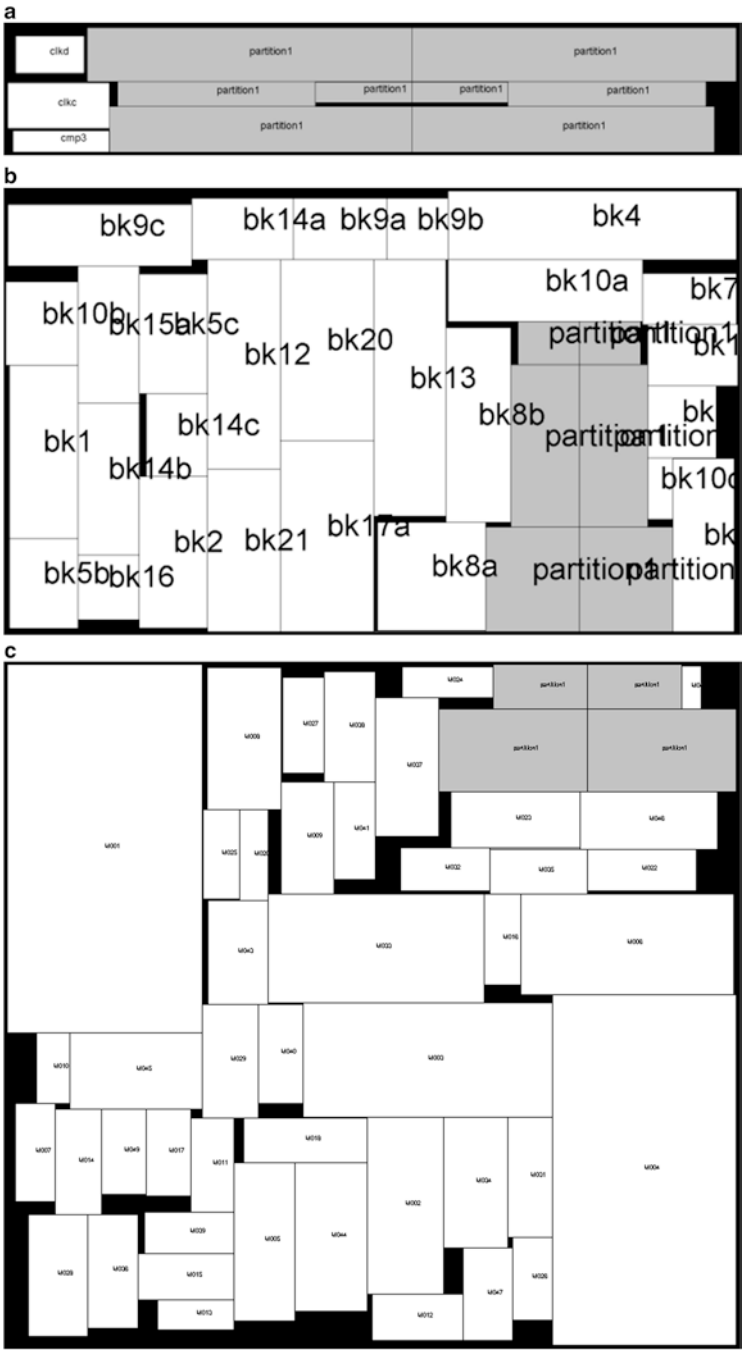


Fig. 8.29 Illustration of the obtained solution that minimizes the placement area: (**a**) *hp*, highlighted solution of Fig. 8.27b, i.e., $f_2(x) = 9.35$ mm^2; (**b**) *ami33*, highlighted solution of Fig. 8.27c, i.e., $f_2(x) = 1.21$ mm^2; and (**c**) *ami49*, highlighted solution of Fig. 8.27d, i.e., $f_2(x) = 38.17$ mm^2. Cells with proximity requirements contained in the proximity group are market at *grey* [18]

Table 8.22 Experimental results for the EM-Aware single-port vs multiport routing approach

Test case		TC7	TC10	TC15	TC25	TC35	TC50	TC75	TC100
#Terminals		7	10	15	25	35	50	75	100
#Sources/#sinks		3/4	4/6	7/8	14/11	18/17	23/27	29/46	55/45
#Ports[a]		[4]	[2–6]	[2–8]	[2–8]	[2–16]	[2–16]	[2–16]	[2–16]
#Obstacles		4	5	6	10	15	15	20	75
#Layers/V_{cost}[b]		4/5	4/5	4/5	4/5	4/5	4/5	4/5	4/5
Single-port/ multiterminal	SP EM-aware WT area[c]	–	29,844	23,436	84,502	162,695	287,191	901,182	1,316,301
	Runtime (s)[d]	–	<0.01	0.02	0.04	0.06	0.15	0.16	1.94
	Rectilinear area[e]	–	29,162	23,454	81,266	163,725	292,269	906,416	1,343,422
	Runtime (s)[d]	–	<0.01	<0.01	0.04	0.03	0.10	0.06	0.21
	Steiner points assigned	–	5	8	20	25	40	84	94
	Runtime (s)[d]	–	<0.01	<0.01	0.04	0.01	<0.01	0.06	0.12
	MP EM-aware WT area[f]	14,200	26,418	21,185	86,601	153,567	292,707	892,346	1,326,902
	Runtime (s)[d]	<0.01	<0.01	0.02	0.04	0.07	0.16	0.22	2.36
Multiport/ multiterminal	MP-selection area[g]	10,820	18,544	14,949	67,280	112,700	216,686	752,994	1,197,635
	Runtime (s)[d]	<0.01	<0.01	<0.01	0.06	0.04	0.21	0.32	0.87
	Steiner points assigned	2	5	5	13	10	20	75	89
	Runtime (s)[d]	<0.01	<0.01	<0.01	0.04	<0.01	<0.01	0.32	0.51
Improvement (%)		–	36	36	17	31	25	17	11

Computational times obtained on an Intel® Core™ i7-3610 @ 2.3 GHz with 8 GB of RAM

[a]Range of number of ports for each terminal

[b]Cost of assigning a via in the global routing process (introduced in Chap. 6 of this book)

[c]Routing area of the EM-aware wiring topology considering single-port terminals

[d]Computational time obtained on an Intel® Core™ i7-3610 CPU 2.3 GHz with 8 GB of RAM

[e]Rectilinear area of the terminal-to-terminal connectivity attained in the wire planning, considering SP/MT

[f]Routing area of the EM-aware wire planning considering MP terminals

[g]Rectilinear area using the multiport selection algorithm (introduced in Chap. 6 of this book)

explore the both the capabilities and flexibility of the proposed methodology, for similar or wide specification changes. A last circuit example, an operational transconductance amplifier was used to show the how the proposed tool efficiency scales for design with more devices. Three benchmark sets were also considered to explore how both the performance and scalability of the optimization-based Placer and fully-automatic Router modules behaves.

When used, electric-currents and current-flow on each terminal had a huge impact on the generated layout. EM-aware WT and current-flow considerations since the beginning of the layout design flow, in the Placer, were shown to greatly increase the layout quality of automatically generated floorplan. Moreover, in the design of the interconnects of current intensive circuits a proper current-flow and EM-reliability is mandatory, as the circuits' performance is heavily compromised if the minimum wire width is applied instead, which often occur when using non-EM-aware methodologies or even in the manual design if the electric-current considerations are ignored. Also, during redesign cycles some electric-currents may change without the corresponding update in the layout. As dealing with a multitude of different currents manually is time-consuming and error-prone, it is not only hard to manually sketch a current-correct wiring topology, but also, any change in the floorplan or in the current-current tree may discard all the previous routing.

During the generation of all layouts throughout this dissertation, over the device routing is allowed. Mainly because AIDA-L's Placers present extremely compact floorplans and for all addressed case studies, over the device wiring did not have a relevant impact on the degradation of the circuit's performance. Obtaining layouts without over the device routing, would require deteriorating the quality of the presented floorplans. However, that feature is supported. In a first phase, in the template-based Placer, routing channels can be used in the construction of the template file to reserve space for routing. While in the optimization-based Placer, the space can be reserved during optimization by computing a straightforward routing resource reservation, where routing channels are obtained with a size proportional to the amount of wiring congestion in that area. For routing, this feature is strongly supported, as the global routing procedure allows to use obstacles during the construction and search on the multilayer multiport grid, and also, during the detailed routing, the electrical-rule check (ERC) procedure can mark, e.g., active area overlaps as a constraint violation for optimization.

Furthermore, AIDA-L offers an innovative solution for parasitic estimation of interconnects by computing the optimal electrical-current correct WT and global routing in-loop for each different sizing solution. The lightweight built-in Parasitic Extractor estimates the impact of layout parasitics for both floorplan and early-stages of routing without requiring a detailed layout, greatly reducing overall evaluation time to in parasitic-aware optimization. Moreover, as the routing template/setup is not fixed, not only the parasitic-aware performances and geometric requirements are considered, but also, the routing topology follows the optimization process. The analog building blocks synthesized using this methodology, shown the generality, accuracy and advantages of avoiding the detailed routing of the proposed approach.

References

1. R. Martins, N. Lourenço, S. Rodrigues, J. Guilherme, N. Horta, AIDA: Automated analog IC design flow from circuit level to layout, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sept 2012, pp. 29–32
2. N. Lourenço, R. Martins, N. Horta, Layout-aware synthesis of analog ICs using floorplan & routing estimates for parasitic extraction, in *Design, Automation & Test in Europe Conference (DATE)*, Mar 2015, pp. 1156–1161
3. R. Martins, N. Lourenço, A. Canelas, R. Póvoa, N. Horta, AIDA: Robust layout-aware synthesis of analog ICs including sizing and layout, in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sept 2015, pp. 1–4
4. R. Castro-Lopez, O. Guerra, E. Roca, F. Fernandez, An integrated layout-synthesis approach for analog ICs. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **27**(7), 1179–1189 (2008)
5. Mentor Graphics, http://www.mentor.com/
6. E. Santin, L. Oliveira, B. Nowacki, J. Goes, A fully integrated and reconfigurable architecture for coherent self-testing of high speed analog-to-digital converters. IEEE Trans. Circ. Syst. I **58**(7), 1531–1541 (2011)
7. K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
8. F. Balasa, K. Lampaert, Symmetry within the sequence-pair representation in the context of placement for analog design. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **19**(7), 721–731 (2000)
9. 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
10. J.-M. Lin, G.-M. Wu, Y.-W. Chang, J.-H. Chuang, Placement with symmetry constraints for analog layout design using TCG-S. Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf. **2**, 1135–1138 (2005)
11. Y.-C. Tam, Y. Young, C. Chu, Analog placement with symmetry and other placement constraints, in *Proceedings of the IEEE/ACM Asia South Pacific Design Automation Conference*, Nov 2006, pp. 349–354
12. 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. Circ. Syst. **33**(6), 879–892 (2014)
13. S. Bandyopadhyay, S. Saha, U. Maulik, K. Deb, A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Trans. Evol. Comput. **12**(3), 269–283 (2008)
14. R. Póvoa, N. Lourenço, N. Horta, R. Santos-Tavares, J. Goes, Single-stage amplifiers with gain enhancement and improved energy-efficiency employing voltage-combiners, in *21st IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Oct 2013, pp. 19–22
15. R. Martins, R. Póvoa, N. Lourenço and N. Horta, Current-flow & current-density-aware multi-objective optimization of analog ic placement. Integr. VLSI J. (in press, 2016). Reprinted from Integration, the VLSI Journal. With permission from Elsevier
16. N. Lourenço, R. Martins, A. Canelas, R. Póvoa, N. Horta, AIDA: layout-aware analog circuit level sizing with in-loop layout generation. Integr. VLSI J. (in press, 2016). Reprinted from Integration, the VLSI Journal. With permission from Elsevier.
17. R. Martins, N. Lourenço, A. Canelas, N. Horta, Electromigration-aware analog Router with multilayer multiport terminal structures. Integr. VLSI J. **47**(4), 532–547 (2014). Reprinted from Integration, the VLSI Journal. With permission from Elsevier
18. R. Martins, N. Lourenco, N. Horta, Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates. Expert Syst. Appl. **42**(23), 9137–9151 (2015). doi:10.1016/j.eswa.2015.08.020. Reprinted from Expert Syst. Appl. With permission from Elsevier

Chapter 9
Conclusions and Future Work

In this book, the proposed methodology and developed tool, AIDA-L, to automate analog integrated circuit (IC) layout generation was presented. This chapter presents the closing remarks, and future directions for the continuous development of AIDA-L.

9.1 Conclusions

AIDA-L progresses in several areas of analog layout design automation and, combines methodologies for placement, routing and parasitic extraction for analog integrated circuits (ICs) in a single tool. The template-based Placer introduces a new level of flexibility by supporting wide changes on device and module's specifications, and also, on the technology design process, while still holding the guidelines provided by the designer. Although the use of non-slicing topological representations, or in particular the B*-tree encoding, is not a novelty, the tool provides a unique specification-independent methodology for the designer to include simple and intuitive high level floorplan guidelines, and, hastily obtain a first cut floorplan solution. In this abstraction layer, the guidelines can be quickly refined by the designer and the resulting floorplan is promptly presented. Also, design reusability and retargetability, once the template file is available, are greatly improved.

On the flip side, in the optimization-based Placer, the concept of hierarchical floorplan automation over an absolute layout representation using multi-objective constrained optimization, implemented by the proposed constrained archive-based multi-objective simulated annealing, is explored. The main weakness pointed to the absolute methods lies in the fact that it may generate infeasible placements with overlapped modules. As overviewed, modeling illegal overlaps as part of a single-objective function requires an increased tuning effort due to the difficulty of predicting an appropriate weight, greatly impairing the performance of absolute representation. However, in the proposed approach, illegal overlaps are dealt as

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9_9

constraints on the constrained multi-objective optimization algorithm, which allows an improved and efficient exploration on unfeasible regions of the solutions space. Furthermore, unlike previous approaches where only one solution is provided, as a result of cost function minimization, the result is here a Pareto front with the trade-off between the desired objectives, e.g., width, length, area, aspect ratio, estimated interconnect length, etc., while the problem's complexity is reduced taking advantage of the hierarchy of the circuits. Unlike most recent approaches, the absolute representation is here explored due to its potential of constraint handling, and also, applying new functionalities to the structure/optimization kernel is almost effortless.

In the fully-automatic Router the solution space is automatically explored using a set of both deterministic and stochastic steps, ensuring that technology design rules are verified, while minimizing the total wiring area and complying with a set of electromigration and IR-Drop constraints. Also, Steiner point and wiring symmetry considerations are taken into account to improve the quality of the solution. Dealing with a multitude of different electric-currents manually and further wiring symmetry is a time-consuming and error-prone task. It is not only hard to manually sketch a symmetric current-correct wiring topology (WT), but even slight changes in the floorplan disposition or in the design specifications may discard all the previous wire planning and/or routing. Furthermore, the proposed multiport multiterminal (MP/MT) routing methodology intends to bypass the limitations of the state-of-the-art approaches, where simplistic 'dot-models' are assumed for the terminals, forcing the inclusion of expert knowledge in the selection of ports, for a proper single-port multiterminal regime. The experimental results showed the advantages of using the developed MP/MT approach to entwine the stages of the automatic Routers with realistic devices' geometries of analog ICs.

AIDA-L's Router outstands from the remaining routing tools presented in the state-of-the-art on analog design automation, by automatically generating flexible routing solutions for any floorplan using only the netlist and a set of electric-currents attached to each terminal, whose results are validated in a commercial tool widely accepted in the industry, Calibre® design rule check (DRC) and layout-versus-schematic (LVS). The introduction of a fully-automatic generation during routing, independently from the circuit's floorplan, allows the designer to explore different floorplans without additional effort. This functionality reinforces the integration of the tool in the bottom-up physical synthesis path of an automatic analog design flow, where any number of different sizing solutions may be attained.

Finally, the Parasitic Extractor improves the computational efficiency of layout-aware methodologies by avoiding the need of a time-consuming detailed layout generation needed by off-the-shelf tools for parasitic extraction, to include parasitic-related data during an automatic layout-aware sizing methodology. Prior knowledge of the circuit's parasitics is not required, i.e., there is no need for circuit specific equations or models. Furthermore, as AIDA-L computes the optimal electrical-current correct WT and global routing in-loop for each different sizing solution, not only the geometric requirements of the floorplan and parasitic-aware performances, but also, the routing quality, will follow the optimization process.

AIDA-L is providing fully functional first cut layout designs, extensively validated in DRC, LVS and with electrical simulations over the extracted netlist, in the presence of complete layout parasitics, to validate the specifications of the original design. Moreover, AIDA-L's layout generation presents a simple, highly reusable and flexible approach for the design automation. The execution time clearly surpass that of handmade design, generating and retargeting traditional analog cells within minutes, and larger examples within hours, a process that could require days, if not weeks, of work when made by an experienced designer.

9.2 Future Work

Analog IC design automation is not a trivial matter, and, although AIDA-L can automatically generate layouts, potential for further developments is still large. This implementation focused on first settling an automatic industrial-level layout generation process, and now it is required to refine that process by addressing more case studies, and adding the insights from experienced analog designers. In the next subsection, the opportunities for future enhancements are outlined.

9.2.1 Improved Efficiency

AIDA-L's development is constantly seeking for improved computational efficiency. While the built-in layout evaluation procedure already features multithreading to speed up the detailed routing, the computation load of the detailed routing and Parasitic Extractor could still get major improved by using a solution of parallel or GPU-accelerated computing. The latest have thousands of cores to process parallel workloads efficiently. As described, AIDA-L encompasses own layout validation and parasitic extraction modules without resorting into commercial tools, this is an advantage when moving to a parallel solution, as the number of licenses is not a limitative factor for the amount of parallelization.

9.2.2 Radio-Frequency

The design of radio-frequency (RF) ICs is one of the most challenging areas in all electronics, due to the fact that parasitic effects of the layout have more impact on the circuit's performance/behavior as the operating frequency increases and as designers go into deeper nanotechnology nodes. RF circuits impose several challenges to the automatic layout generation, as the matter of fact this subject is barely addressed in the literature. AIDA-C is already sizing RF blocks such as LC-Oscillators, LC-Voltage Controlled Oscillators, narrowband differential

low-noise amplifiers and mixers, and AIDA-L should follow. The inclusion of an extended analog module generator that features RF models for the transistors, capacitors, resistances and inductors is mandatory. Furthermore, 45° considerations during routing could approximate the solutions from the designers' needs, and the severe impact from parasitics will create significant challenges to the Parasitic Extractor.

9.2.3 Deep-Nanometer Technologies

The United Microelectronics Corporation 130 nm technology design kit used through this book accompanied the development of AIDA-L. While the 130 nm node is still widely used for manufacturing analog ICs in both industry and academic realities, the endnotes of future developments shall be the improvements required to AIDA-L in order to be compliant with deeper nanometer technologies, e.g., 90 nm and 65 nm. While, e.g., the 28 nm-nodes will represent a completely new reality in terms of integration. The inclusions of more specific electrical constraints, e.g., well proximity effect or length of oxide diffusion, are just some examples of the challenges that have to be considered when dealing with smaller design processes.

Index

R. Martins et al., *Analog Integrated Circuit Design Automation*,
DOI 10.1007/978-3-319-34060-9