Analog Layout Synthesis
Analog Layout Synthesis

A Survey of Topological Approaches
Analog components appear on 75% of all chips, and cause 40% of the design effort and 50% of the design errors detected after first silicon measurements, reported *EDA Weekly* on March 21, 2005. Due to increasing functional complexity of system-on-chips, the difficulties in analog design and the lack of design automation support for analog circuits continually increase the bottleneck character of analog components in chip design. Design methodology and design automation for analog circuits therefore is a crucial problem for future system-on-chips.

Eminently critical is the layout synthesis part of the analog design flow. Although there have been a lot of very good works from universities over the years, some of which even found their way to commercial EDA tools, industrial application of analog layout synthesis is still in its infancy when it is compared to its digital counterpart! The industrial point of view even says that practicable EDA tools for analog layout synthesis did not exist.

But it seems that this situation is about to change. In the face of increasing circuit complexity and high performance SoC designs, the once-sleepy analog EDA market is experiencing an increasing shift from single vendor solutions to design tool integration via alliances between many players. The attempt to create an inter-platform reference, such as the Interoperable PDK Libraries (IPL) alliance, where analog layouts made with a tool can be imported error-free to different frameworks, is an example. Many EDA start-ups as well as major leaders are already announcing key automated layout tools for the analog designer intended to boost his/her productivity.

In this exciting scenario, academia continues to strive for new, more efficient, and complementary approaches to this task and to the existing tools, and has recently produced some very interesting new solutions. The intention of this book has two parts. On the one hand, it summarizes and presents these latest results. On the other hand, it is dedicated to give an introduction to advanced analog layout methods on the graduate level.

The book is structured in three parts. The first part with three chapters covers recent approaches to topological placement of analog circuits. The second part treats the problem of routing. The third part with three more chapters deals with layout in the design flow, namely, with the problem of retargeting an existing layout for a new technology, with integrating layout in the sizing process, and with constraint management in the design flow.
The first chapter starts with an introduction to the different ways of approaching in CAD tools device-level placement problems for analog layout. It is elaborated how the structural representation of the layout in the algorithm is crucial for the efficiency and efficacy of the placement process. Besides the classical way of using absolute coordinates for the module placement and slicing structures for topological representations, which encode the relative positioning between cells, it describes how the sequence-pair and tree-based topological representation can be applied to dramatically reduce the search space to the tiny fraction, which satisfies the inherent symmetry constraints in analog circuits. It further develops sufficient conditions to ensure the symmetry constraints during the successive moves of a placement algorithm and, based on these ideas, presents several topological algorithms that perform the exploration process very efficiently.

The second chapter furthers the ideas presented in the first chapter and extends them to a hierarchical module clustering. The analog devices can be hierarchically clustered into groups according to models, circuit functionalities, or signal/current flows. Following the B*-tree, a hierarchical B*-tree (HB*-tree) placement representation is developed to model this circuit hierarchy and symmetry and proximity constraints among modules and across the hierarchy. This hierarchical representation is fed into a placement algorithm to generate optimum device placements that meet all device layout constraints. Performing a simulated annealing algorithm, the placement of the device modules in different device groups belonging to different clustering hierarchies is simultaneously optimized.

The third chapter first introduces a method to automatically derive the circuit hierarchy and the resulting symmetry, proximity, and matching constraints from a netlist. A deterministic algorithm is then presented that computes the shape function of different aspect ratios of the circuit placement by a recursive bottom-up approach through the derived circuit hierarchy starting from basic modules such as current mirrors or differential pairs. For each hierarchy level, the shape function is determined by combining the placements of the next-lower hierarchy. These are stored as so-called enhanced shape functions that include the corresponding B*-trees of each individual shape. Algorithms are proposed to generate the vertical and horizontal sum of two B*-Trees of placements while provably complying with the constraints. As the algorithm bounds the enumeration according to the circuit hierarchy and the constraints, it generates results very fast, while being deterministic without any tuning parameter.

The second part of the book deals with analog routing. It gives a tutorial on routing methods and corresponding placement and routing representations, including constraints, for instance, for symmetry or crosstalk. A review of different routing strategies and the corresponding state of the art follows. Early routing approaches inspired from digital design, cost-driven approaches, and parasitic-driven approaches (including, e.g., performance sensitivities), as well as the A* algorithm are covered. The connection to placement through templates and other integration approaches is discussed afterward. Then, the partitioning of routing into global and detailed routing, as in digital design, is described. The chapter concludes with specialized routing approaches for RF circuits and analog arrays.
The third part of the book addresses analog layout issues arising from the ambient design flow.

In Chap. 5, the task of retargeting an existing layout, including placement and routing, is examined. Specific algorithms for layout retargeting may be beneficial if the involved layout modifications are moderate or to extract and conserve the knowledge contained in a layout. After a short introduction to the preparatory steps of layer mapping, constraint generation and device recognition, the main algorithmic step of retargeting, i.e., layout compaction, is described in detail. Based on the linear programming approach to its solution, a graph-based simplex method is presented with full details. The different types of constraints, the complexity of the algorithm, and practical issues are discussed as well.

Chapter 6 is dedicated to the problem of integrating layout effects into the circuit sizing process, to avoid unnecessary iterations between electrical and physical synthesis as much as possible. This has been called parasitic-aware synthesis. This chapter reaches from the very basics (what is it, and why and when is it really necessary) to a practical implementation of this type of synthesis process. Different methods to carry it out as well as their pros, cons, and trade-offs (mainly efficiency vs. completion time) will be explained. A technique will be presented that uses a combination of simulation-based optimization, procedural layout generation, exhaustive geometric evaluation algorithms, and several mechanisms for parasitic estimation, to comprehensively incorporate the layout-induced parasitic into electrical synthesis.

Chapter 7 concludes the book with a discussion of the management of the crucial factor in analog layout — the constraints. It provides a problem formulation for the classification, representation, transformation, and verification of constraints in a top-down design flow, as well as a formulation of a constraint engineering system, including its impact on the design flow and its algorithms.

This bow from placement to routing to the design flow, drawn by the structure of the book, invites the reader to start from the beginning and read one chapter after the other. At the same time, the chapters are self-contained and may be accessed individually and independently. In any way she or he approaches the book, the reader will gain a deep insight into the tasks of analog layout and into the actual solution approaches.

Munich

Helmut Graeb

March 2010
The Authors

Hazem Abbas *(Hazem_Abbas@mentor.com)* received the B.Sc. and M.Sc. degrees in 1983 and 1988, respectively, from Ain Shams University, Egypt and the Ph.D. degree in 1993 from Queen’s University at Kingston, Canada all in Electrical and Computer Engineering. He held a postdoc position at Queen’s University in 1993. In 1995, he worked as Research Fellow at the Royal Military College at Kingston and then joined the IBM Toronto Lab as a Research Associate. He joined the Department of Electrical and Computer Engineering at Queen’s University as an Adjunct Assistant Professor in 1997–1998. He is now with the Department of Computers and Systems Engineering at Ain Shams University, Egypt, as a Professor on Computer and Systems Engineering. Dr. Abbas is also working with Mentor Graphics Inc., Egypt as a Senior Engineering Manager. His research interests are in the areas of neural networks, pattern recognition, evolutionary computations, and image processing and their parallel and multicore implementations. He also serves as the President of the IEEE Signal Processing Chapter in Cairo.

Florin Balasa *(balasa@suu.edu)* received the Ph.D. degree in computer science from the Polytechnical University of Bucharest, Bucharest, Romania, in 1994, and the Ph.D. degree in electrical engineering from the Katholieke Universiteit Leuven, Leuven, Belgium, in 1995.

From 1990 to 1995, he was with the Interuniversity Microelectronics Center (IMEC), Leuven, Belgium. From 1995 to 2000, he was a Senior Design Automation Engineer at the Advanced Technology Division of Conexant Systems (formerly Rockwell Semiconductor Systems), Newport Beach, CA. He is currently an Associate Professor of Computer Science at the Southern Utah University.

Dr. Balasa was a recipient of the US National Science Foundation CAREER Award.

Rafael Castro-López *(castro@imse-cnm.csic.es)* received the “Licenciado en Física Electrónica” degree (M.S. degree on Electronic Physics) and the “Doctor en Ciencias Físicas” (Ph.D. degree) from the University of Seville, Spain, in 1998 and 2005, respectively. Since 1998, he has been working at the Institute of Microelectronics of Seville (CSIC-IMSE-CNM) of the Spanish Microelectronics Center, where he now holds the position of Tenured Scientist. His research interests lie in the field of integrated circuits, especially design and computer-aided design
for analog and mixed-signal circuits. He has participated in several national and international R&D projects and co-authored more than 50 international scientific publications, including journals, conference papers, book chapters, and the book *Reuse-based Methodologies and Tools in the Design of Analog and Mixed-Signal Integrated Circuits* (Springer, 2006).

**Yao-Wen Chang** (ywchang@cc.ee.ntu.edu.tw) received the B.S. degree from National Taiwan University in 1988, and the M.S. and Ph.D. degrees from the University of Texas at Austin in 1993 and 1996, respectively, all in computer science.

Currently, he is Professor of the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan. His current research interest lies in electronic design automation, with an emphasis on physical design for nanometer IC’s and design for manufacturability. He has co-edited one textbook on Electronic Design Automation and co-authored one book on routing and more than 180 ACM/IEEE conference/journal papers in these areas. Dr. Chang is a four-time winner of the ACM ISPD contests (1st and 3rd places in the respective 2009 and 2010 Clock Network Synthesis Contests, second place in the 2008 Global Routing Contest, and third place in the 2006 Placement Contest). Dr. Chang received seven excellent teaching awards, Best Paper Awards at the 2007 and 2008 VLSI Design/CAD Symposia, a Best Paper Award at ICCD-95, and 14 Best Paper nominations from DAC (four times), ICCAD (three times), ISPD (four times), TODAES, ASP-DAC, and ICCD.

Dr. Chang is currently an associate editor of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) and an editor of the Journal of Information Science and Engineering (JISE). He has served as the ACM ISPD General Chair, and the ACM ISPD and IEEE FPT Technical Program Chairs, on the IEEE/ACM ICCAD Executive Committee, the IEEE/ACM ASP-DAC Steering Committee, the ACM/SIGDA Physical Design Technical Committee, the IEEE CEDA Conferences Committee, and the technical program committees of major EDA conferences, including ASP-DAC, DAC, DATE, FPL, FPT, GLSVLSI, ICCAD, IECON, ISPD, SOCC, TENC0N, and VLSI-DAT. He has served as the chair of the EDA Consortium of the Ministry of Education of Taiwan, Principal Reviewer of the SBIR Projects of the Ministry of Economic Affairs, Review and Planning Committee Member of the National Science Council of Taiwan, Independent Board Director of Genesys Logic, Inc., Technical Consultant of Faraday Technology Inc., MediaTek Inc., and RealTek Semiconductor Corp., and Member of Board of Governors of Taiwan IC Design Society.

**Mohamed Dessouky** (Mohamed.Dessouky@mentor.com) received the Ph.D. degree in electrical engineering from the University of Paris VI, France, in 2001. In 1992, he joined the Electronics and Electrical Communications engineering department, University of Ain Shams, Egypt, where he was a Research and Teaching Assistant, and is currently an Associate Professor. Since 2004, he is on leave to Mentor Graphics Inc., Egypt, where he served as the head of the analog design team.
in the IP Division and is currently a Staff Engineer. His research interests include the design of switched-capacitor circuits, analog-to-digital conversion and CAD for analog, RF and mixed-signal integrated circuits.

Güngör Dündar (dundar@boun.edu.tr) got his B.S. and M.S. degrees from Boğaziçi University, Istanbul, Turkey in 1989 and 1991, respectively, and his Ph.D. degree from Rensselaer Polytechnic Institute in 1993, all in electrical engineering. Since 1994, he has been with the Department of Electrical and Electronic Engineering, Boğaziçi University, where he is currently a professor, with some temporary positions at the Turkish Naval Academy, EPFL (Lausanne, Switzerland), and the Technical University of Munich during this period.

Dr. Dündar has published more than 100 papers in international journals and conferences and a book on analog design automation. During his career, he has received various awards, among which are the best paper award in the IEEE ASAP conference in 2008 and the Turkish Scientific and Technological Council Encouragement Award for Research in 2009. His research interests include analog and mixed signal integrated circuit design and design automation, especially for analog circuits.

Michael Eick (michael.eick@mytum.de) received the Dipl.-Ing. Degree (equivalent to M.Sc.) in Electrical Engineering and Information Technology from Technische Universität München, Munich, Germany in 2008. Since 2008, he is working as a research and teaching assistant at the Institute for Electronic Design Automation, Technische Universität München, Munich, Germany. He is currently working toward his Ph.D. His research interests cover the automatic analysis of circuit structures of analog circuits with application to layout and sizing.

Reem El-Adawi (Reem_ElAdawi@mentor.com) received the B.Sc. and M.Sc. degrees in 1993 and 2001, respectively, from Ain Shams University, Egypt. Since 1995, she is with Mentor Graphics Inc., Egypt, where she has held various positions. She is currently Engineering Manager for the team responsible for the development of Chameleon ART, an analog retargeting tool.

Francisco V. Fernández (pacov@imse-cnm.csic.es) got the Physics-Electronics degree from the University of Seville in 1988 and his Ph.D. degree in 1992. In 1993, he worked as a postdoctoral research fellow at Katholieke Universiteit Leuven (Belgium). From 1995 to 2009, he was an Associate Professor at the Dept. of Electronics and Electromagnetism of University of Seville, where he was promoted to full professor in 2009. He is also a researcher at CSIC-IMSE-CNM. His research interests lie in the design and design methodologies of analog and mixed-signal circuits. Dr. Fernández has authored or edited three books and has co-authored more than 100 papers in international journals and conferences. Dr. Fernández is currently the Editor-in-Chief of Integration, the VLSI Journal (Elsevier). He regularly serves at the Program Committee of several international conferences. He has also participated as researcher or main researcher in several National and European R&D projects.

Jan Freuer (jan.freuer@de.bosch.com) received his diploma degree in Computer Science from the Eberhard-Karls University in Tübingen, Germany in 2002. Since
2004, he has been with the R&D department AE/EIM of the automotive electronics division of Robert Bosch GmbH in Reutlingen, Germany. He is currently finalizing his Ph.D. degree in computer science at the Carl von Ossietzky University in Oldenburg, Germany. His main research interests are related to constraint-engineering topics for analog design automation.

Helmut Graeb (graeb@tum.de) got his Dipl.-Ing., Dr.-Ing., and habilitation degrees from Technische Universität München in 1986, 1993, and 2008, respectively. He was with Siemens Corporation, Munich, from 1986 to 1987, where he was involved in the design of DRAMs. Since 1987, he has been with the Institute of Electronic Design Automation, TUM, where he has been the head of a research group since 1993.

He has published more than 100 papers, six of which were nominated for best papers at DAC, ICCAD, and DATE conferences. His research interests are in design automation for analog and mixed-signal circuits, with particular emphasis on Pareto optimization of analog circuits considering parameter tolerances, analog design for yield and reliability, hierarchical sizing of analog circuits, analog/mixed signal test design, semidiscrete optimization of analog circuits, structural analysis of analog and digital circuits, and analog placement.

Dr. Graeb has, for instance, served as a Member or Chair of the Analog Program Subcommittees of the ICCAD, DAC, and D.A.T.E conferences, as Associate Editor of the IEEE Transactions on Circuits and Systems Part II: Analog and Digital Signal Processing and IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, and as a Member of the Technical Advisory Board of MunEDA GmbH Munich. He is Senior Member of IEEE (CAS) and member of VDE (ITG).

He was the recipient of the 2008 prize of the Information Technology Society (ITG) of the Association for Electrical, Electronic, and Information Technologies (VDE), of the 2004 Best Teaching Award of the TUM EE Faculty Students Association, of the 3rd prize of the 1996 Munich Business Plan Contest.

Göran Jerke (goeran.jerke@ieee.org) received his diploma degree in Electrical Engineering from the Dresden University of Technology in Dresden, Germany. Since 2000, he is with the R&D department AE/EIM of the automotive electronics division of Robert Bosch GmbH in Reutlingen, Germany. His research work focuses on analog design automation and reliability-aware physical design of ICs where he has published numerous papers. In 2004, he received the EDA Achievement Award from the German edacentrum e.V. He is currently finishing his Ph.D. degree in Electrical Engineering at the Dresden University of Technology.

Jens Lienig (jens.lienig@ifte.de) received the M.Sc. (diploma), Ph.D. (Dr.-Ing.) and habilitation degrees in Electrical Engineering from Dresden University of Technology, Dresden, Germany, in 1988, 1991, and 1996, respectively. He is currently a Full Professor of Electrical Engineering at Dresden University of Technology, where he is also Director of the Institute of Electromechanical and Electronic Design. From 1999 to 2002, he worked as Tool Manager at Robert Bosch GmbH in Reutlingen, Germany, and from 1996 to 1999, he was with Tanner Research Inc. in
Pasadena, CA. From 1994 to 1996, he was a Visiting Assistant Professor with the Department of Computer Science, University of Virginia, Charlottesville, VA. From 1991 to 1994, he was a Postdoctoral Fellow at Concordia University in Montréal, PQ, Canada. His current research interests are in the physical design automation of VLSI circuits, MCMs, and PCBs, with a special emphasis on electromigration, 3D design, and constraint-driven design methodologies. Prof. Lienig has served on the Technical Program Committees of the DATE, SLIP, and ISPD conferences. He is a Senior Member of IEEE.

Mark Po-Hung Lin (marklin@ccu.edu.tw) received the B.S. and M.S. degrees in Electronics Engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1998 and 2000, respectively, and the Ph.D. degree in the Graduate Institute of Electronics Engineering (GIEE), National Taiwan University (NTU), Taipei, Taiwan, in 2009.

From 2000 to 2007, he was with Springsoft, Inc., Hsinchu, Taiwan, where he was involved in the design of automatic schematic generation in both VerdiTM Automated Debug System and LakerTM Advanced Design Platform, and the design of custom placement and routing in LakerTM Custom Layout System. In 2008, he was a visiting scholar in the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign (UIUC), IL, USA. He is currently an Assistant Professor of the Department of Electrical Engineering, National Chung Cheng University, Chiayi, Taiwan. His current research interest lies in electronic design automation, with an emphasis on physical design for analog IC’s.

Elisenda Roca (eli@imse-cnm.csic.es) received the physics and the Ph.D. degrees from the University of Barcelona, Spain, in 1990 and 1995, respectively. From November 1990 to April 1995, she worked at IMEC, Leuven, Belgium, in the field of infrared detection aiming to obtain large arrays of CMOS compatible silicide Schottky diodes. Since 1995, she has been with the Institute of Microelectronics of Seville, (IMSE-CNM-CSIC), Spain, where she holds the position of Tenured Scientist. Her research interests are centered in the design of CMOS Vision Systems on-Chip (VSoC) with integrated high dynamic range sensors, and modeling and design methodologies for analog integrated circuits. She has been involved in several research projects with different institutions: Commission of the EU, ESA, ONR-NICOP, etc. She has also co-authored more than 50 papers in international journals, books, and conference proceedings.

Hazem Said (Hazem.Said@eng.asu.edu.eg) received the B.Sc. and M.Sc. degrees in Computer Engineering from Ain Shams University, Egypt, in 1999 and 2006, respectively. He is currently pursuing his Ph.D. degree in electrical engineering at the same University in the field of fast electrical simulation. He worked as a Senior Development Engineer at Mentor Graphics Inc., Egypt, on automatic analog design migration from 2002 to 2007. His research interests include Computer Algorithms, Electronic CAD Algorithms, Artificial Intelligence, and Quantum Computing.

Ulf Schlichtmann (ulf.schlichtmann@tum.de) received the Dipl.-Ing. and Dr.-Ing. degrees in Electrical Engineering and Information Technology from Technische
Universität München (TUM), Munich, Germany, in 1990 and 1995, respectively. From 1994 to 2003, he was with the Semiconductor Group of Siemens AG which, in 1999, became Infineon Technologies AG. There he held various technical and management positions in design automation, design libraries, IP reuse, and product development. Since 2003, he has been with TUM as Professor and Head of the Institute for Electronic Design Automation. His research interests are in computer-aided design of electronic circuits and systems, with special emphasis on designing robust systems. Since 2008, he serves as Dean of TUM’s department of Electrical Engineering and Information Technology.

Hussein Shahein (hussein.shahin@eng.asu.edu.eg) obtained his Ph.D. from the Moore School of Electrical Engineering, University of Pennsylvania, PA, USA, in 1972. He was a postdoctoral Fellow with the University of Pennsylvania Computer Center (Uni-Coll). He also worked at the IBM research Laboratories in San Jose, CA, USA. He then joined the Computer and Systems Engineering Department, Ain Shams University, Egypt. Prof. Shahein served as visiting Professor at King AbdulAziz University, Jeddah, KSA, the Computer Science Dept., University of Bahrain, and the Electrical Engineering Department, UAE University, Al Ain, where he was the Department Chair. He also served as the Chair of the Computer and Systems Engineering Department, Ain Shams University, Egypt, and the Dean of the Faculty of Computer Science, Misr International University, Egypt. He is the Chairman of the Board of Elmohandis Information Systems Company. Prof. Shahein has supervised many Ph.D. and M.Sc. students. His research interests include Optimization, Computer Networks, Networks and data Security.

Martin Strasser (strasser@tum.de) received his Dipl.-Ing. degree in Electrical Engineering and Information Technology from Technische Universität München (TUM), Munich, Germany, in 2005. From 2003 to 2005, he was with MunEDA GmbH, before he joined the Institute for Electronic Design Automation at TUM. In this institute, he is currently working toward his Dr.-Ing. degree. His research interests are in electronic design automation for analog circuits, especially the automation of the layout generation.

Ahmet Unutulmaz (unutulm@boun.edu.tr) received his B.S. and M.S. degrees in Electrical Engineering from Boğaziçi University, Istanbul, Turkey in 2008. Since 2008, he has been working as a research and teaching assistant at the Department of Electrical Engineering, Boğaziçi University, Istanbul, Turkey. His research interests include automatic synthesis of analog layouts and integration of design automation tools.
## Contents

### Part I  Placement

1. **Device-Level Topological Placement with Symmetry Constraints**
   Florin Balasa ................................................................. 3

2. **Hierarchical Placement with Layout Constraints**
   Mark Po-Hung Lin and Yao-Wen Chang .................. 61

3. **Deterministic Analog Placement by Enhanced Shape Functions**
   Martin Strasser, Michael Eick, Helmut Graeb, and Ulf Schlichtmann 95

### Part II  Routing

4. **Routing Analog Circuits**
   Günhan Dündar and Ahmet Unutulmaz 149

### Part III  Layout in the Design Flow

5. **Analog Layout Retargeting**
   Hazem Said, Mohamed Dessouky, Reem El-Adawi, Hazem Abbas, and Hussein Shahein 205

6. **Closing the Gap Between Electrical and Physical Design: The Layout-Aware Solution**
   Rafael Castro-López, Elisenda Roca, and Francisco V. Fernández 243

7. **Constraint-Driven Design Methodology: A Path to Analog Design Automation**
   Göran Jerke, Jens Lienig, and Jan B. Freuer 269

### Index

................................................................................. 299
Part I
Placement
Chapter 1
Device-Level Topological Placement with Symmetry Constraints

Florin Balasa

Abstract The traditional way of approaching placement problems in computer-aided design (CAD) tools for analog layout is to explore an extremely large search space of feasible or unfeasible placement configurations (called flat representations of the layout), where the cells are moved in the chip plane by a stochastic optimizer – like simulated annealing or a genetic algorithm.

This chapter discusses the possible use in analog placement problems with symmetry constraints of topological representations of the layout, encoding systems that are not restricted to slicing floorplan topologies. First, the chapter gives an overview of several data structures that may be used in the evaluation of various topological representations of the layout – therefore, in building the placement from the layout encoding. Afterwards, the chapter presents a subset of sequence-pairs – called “symmetric-feasible” – that allows to take into account the presence of an arbitrary number of symmetry groups of devices during the exploration of the solution space. Alternatively, the possible use of tree representations instead of “symmetric-feasible” sequence-pairs is also discussed.

The computation times exhibited by the topological approaches are significantly better than those of the placement algorithms using the traditional exploration strategy based on flat representations, while preserving a similar quality of the placement solutions.

1.1 Introduction

1.1.1 CAD for Analog Layout

In recent years, complete systems that used to occupy one or more boards have been integrated on a few chips or even on a single chip. Examples of such systems-on-a-chip (SoC’s) are networking interfaces, wireless designs, or new
generations of integrated telecommunication systems – that include analog, digital, and eventually, radiofrequency (RF) sections on one chip. Although most functions in such integrated systems are implemented with digital or digital signal processing circuitry, the analog circuits needed at the interface between the electronic system and the real world are now being integrated on the same die for reasons of cost and performance.

In the digital domain, computer-aided design (CAD) tools are fairly well developed, especially for the lower level of the design flow. Unlike analog circuits, a digital system can naturally be modeled in terms of Boolean representations and programming language constraints; its functionality can easier be represented in algorithmic form. Consequently, many lower-level aspects of the digital design process are fully automated. Research interests are now moving in the direction of system synthesis, where system-level specifications are translated into hardware–software co-architecture. The level of automation is far from the “push-button” stage, but the advance of CAD tools is keeping up reasonably well with the progress of technology.

Unfortunately, the situation is worse on the analog side. Apart from circuit simulators, layout editing environments, or layout verification tools, real commercial solutions are only beginning to appear as the result of a valuable research and development (R&D) effort in the field [1–3]. Some of the main reasons for this lack of automation are that analog design in general is less systematic and more heuris-tic in nature than digital design, requiring specialized knowledge, design skills, and years of experience; analog circuits are more sensitive to parasitic disturbances, crosstalk, substrate noise, supply noise, etc.; in addition, the variety of schematics and diversity of device sizes and shapes are much larger. These differences from digital design explain why specific analog solutions need to be developed. Due to the lack of mature, robust analog CAD tools, analog designs today are still largely being handcrafted, with limited CAD support available (except simulators, interactive layout environments). The design cycle for analog (and mixed-signal) IC’s remains long and error prone.

The physical implementation step in the analog design flow corresponds to a variety of tasks that can be grouped into two major areas: (a) analog circuit-level (or block-level) layout synthesis, which has to transform a sized transistor-level schematic into a mask layout, and (b) system-level layout assembly, in which the basic functional blocks are already laid out and the goal is to floorplan, place, and route them, as well as to distribute the power and ground connections. These two areas are also interleaved as most design flows require a mix of top-down and bottom-up approaches. This chapter will address placement issues in the field of block-level layout synthesis.

The optimization-based place-and-route layout generation approaches consist of synthesizing the layout solution by optimization techniques according to some cost functions. They differ from the earlier procedural module generation techniques [4], in which the layout of the entire circuit is precoded in a software tool that generates the complete layout for the actual parameter values entered at run time. Also, they differ from the related set of template-driven methods [5], where a geometric
template fixing the relative position and interconnection of the devices is stored for each circuit. The advantages of the optimization-based approaches are their generality and flexibility in terms of performance and area. The penalty to pay is they require a more significant computational effort; also, the layout quality is more dependent on the algorithms, on the cost functions employed, on providing a complete set of design constraints and taking them into account during the optimization.

### 1.1.2 The Device-Level Analog Placement Problem

The decision whether a given set of fixed-oriented rectangles, having widths and heights real numbers, could be packed onto a chip of known width and height was proven to be NP-complete [6], while the problem of finding a minimum area packing was shown to be NP-hard. Like many other VLSI placement problems – for instance, chip floorplanning and macro cell digital placement – the analog placement must also cope with optimally packing arbitrarily sized modules.

In addition to that, a placement tool must include specific capabilities to automatically produce analog device-level layouts matching in density and performance the high-quality manual layouts. Such specific features are, for instance, (1) the ability to deal with topological constraints for symmetry and device matching; (2) the ability to arrange devices such that critical structures are shared – design technique known as device merging or geometry sharing [7], aiming to reduce both layout density and induced parasitics; (3) the existence of a (built-in) library of predefined module generators and the ability to exploit their reshaping capabilities during the placement process [1].

### 1.1.3 Overview of Analog Placement Methods

Due to the complexity of the basic problem, several heuristic placement techniques have been attempted first. The constructive approaches consist in evolving gradually the placement solution by selecting one module at a time and positioning it in the “best” available location. Several systems for analog placement employed constructive methods: Kayal et al. developed an expert knowledge base to guide the placement [8]; Mehranfar suggested a schematic-driven approach, using a constructive scheme based on connectivity and relative positioning in the input schematic [9, 10]. The constructive methods are fast, scaling well with the problem size; their basic drawback is the dependence on the selection order of devices. Lacking a global view in dealing with a variety of interacting quality measures, this strategy yields sometimes poor placement solutions. A technique achieving a better global optimization of the device positions – by iteratively combining min-cut partitioning and force-directed placement – has been employed in an interactive environment for full-custom designs [11].
Other class of methods translates an analog placement problem into a constrained (combinatorial) optimization. Earlier techniques extracted mainly (hard and soft) nonquantitative constraints for the subsequent optimization phase [12]. In later approaches, the optimization was performance-driven, doing a quantitative evaluation (based on estimation models) of the placement solutions, to ensure the performance of the final layout [13, 14].

As combinatorial optimization engines, the simulated annealing [15] and genetic algorithms [16] were effective choices for solving industrial analog placement problems. These algorithms use stochastically controlled “hill-climbing” to avoid being trapped in local minima during the optimization process. In addition, they do not impose severe constraints on the size of the problems or on the mathematical properties of the cost function – like most optimization algorithms in mathematical programming. While efficiently trading-off between a variety of layout factors – such as area, total net length, aspect ratio, maximum chip width and/or height, cell orientation, “soft” cell shape, etc. – they support incremental addition of new functionality (for instance, updates of cost function and/or constraints) and they are relatively easy to implement (although good tuning needs more time). This is why simulated annealing, the most mature of the stochastic techniques, provided the engine for effective software packages both in digital (TimberWolfSC v7.0 [17]) and in analog design: ILAC [18], KOAN/ANAGRAM II [7, 12] – that evolved into the NeoLinear system, PUPPY-A [13], LAYLA [14]. More recently, a two-phase approach using both a genetic algorithm and simulated annealing with dynamic adjustment of the parameters has been reported [19, 20]. Another recent technique derives linear inequalities from constraint graphs extracted from sequence-pairs, and obtains the placement by linear programming within a simulated annealing framework [21].

While this chapter will focus on optimization techniques for analog placement, other effective analog layout tools are template-driven [22, 23]. These tools are built on template databases containing analog circuits designed by experienced experts. Upon the arrival of a new design demand, the system selects a suitable template from the database, adding information on the target technology, design rules, device sizes, etc., to re-generate automatically the target layout.

### 1.1.4 Placement for Layout Symmetry

In high-performance analog circuits, it is often required that groups of devices are placed symmetrically with respect to one or several axes. Differential circuit techniques are used extensively to improve the accuracy, power supply rejection ratio, and dynamic range of many analog circuits. The full performance potential of many of these circuits cannot be achieved unless special care is taken to match the layout parasitics in the two halves of the differential signal path. Failure to match these parasitics in, for instance, differential analog circuits can lead to higher offset voltages.
and degraded power-supply rejection ratio [7]. The main reason of symmetric placement (and routing, as well) is to match the layout-induced parasitics in the two halves of a group of devices.

Placement symmetry can also be used to reduce the circuit sensitivity to thermal gradients. Some VLSI devices (the bipolar devices, in particular) exhibit a strong sensitivity to ambient temperature. If two such devices are placed randomly relative to the isothermal lines, a temperature-difference mismatch may result. Failure to adequately balance thermal couplings in a differential circuit can even introduce unwanted oscillations [24]. To combat potentially induced mismatches, the thermally sensitive device couples should be placed symmetrically relative to the thermally radiating devices. Since the symmetrically placed sensitive components are equidistant from the radiating component(s), they see roughly identical ambient temperatures and no temperature-induced mismatch results.

It is more often the case that a circuit has a mix of symmetric and asymmetric components. For example, the two-stage Miller compensated opamp shown in Fig. 1.1 has a symmetric differential input stage, but it has an asymmetric single-ended output stage.

The typical forms of symmetry which should be handled by an analog placement tool are [7]:

1. Mirror symmetry: Consists in placing a symmetry group of cells about a common axis such that the cells in every pair have identical geometry and mirror-symmetric orientation. It is the most standard form of layout symmetry. There are two major advantages of this placement arrangement. First, because sibling devices are forced to adopt identical geometry, device-related

Fig. 1.1 Schematic of a two-stage Miller compensated opamp with asymmetric output stage
parasitics are balanced and device matching characteristics are improved. Second, mirror-symmetric placement aligns device terminals in a way that makes mirror-symmetric routing\textsuperscript{1} possible.

2. Perfect symmetry: Differs from the previous by the identical (rather than mirror-symmetric) orientations of the paired devices. This type of symmetry is sometimes required in order to meet very stringent matching requirements. When there is a possibility of anisotropic fabrication disturbances (e.g., oblique-angle ion implantation) \cite{7}, the best matching is achieved when paired devices are placed in identical orientations. Perfectly symmetric placement presents a difficult layout problem: because the device terminals are no longer mirror-symmetric, one cannot use mirror-symmetric routing to connect sibling devices with parasitic matched wires. Instead, one has to route parasitic matched wires, which are not geometrically symmetric. This can be particularly difficult when there is a mix of symmetric and asymmetric circuitry.

3. Self-symmetry: Characteristic for devices presenting a geometrical symmetry and sharing the same axis with other pairs of symmetric devices. Self-symmetric devices have two uses. First, it is often desirable to place asymmetric devices (e.g., devices in bias networks) in the middle of a mirror-symmetric layout. This greatly simplifies wiring in the case that the device is highly connected to devices on both sides of the symmetric signal path. Such an arrangement presents mirror-symmetric terminals to the left and right halves of the circuit, so that they can participate in mirror-symmetric routing. Second, self-symmetry is useful in creating thermally symmetric layouts.

A subset of cells is called a symmetry group if all cells are exhibiting a form of symmetry and, in addition, they all share a common symmetry axis. The symmetry constraints for a pair of devices \((B_i, B_j)\) in the \(k\)th symmetry group have the form: \((x_i + w_i) + x_j = 2 \cdot x_{\text{symAxis}_k}\) and \(y_i = y_j\), where \((x_i, y_i)\) are the left-bottom coordinates of device \(B_i\), \(w_i\) denotes its width, and \(x_{\text{symAxis}_k}\) is the abscissa of the symmetry axis of the \(k\)th group (assuming the axis is vertical). Similarly, a self-symmetric device \(B_i\) must satisfy the constraint: \(x_i + w_i/2 = x_{\text{symAxis}_k}\). In this chapter, the symmetry axes will be considered vertical since this is the typical way most layouts are designed.

\subsection{1.1.5 The Absolute Representation of the Layout}

A combinatorial optimization algorithm for solving placement problems can equally operate with two distinct spatial representations of the placement configurations. The earliest is the so-called absolute (or flat) representation introduced by Jepsen and Gellat \cite{25} in a macro-cell placement tool. In this representation, the cells

\textsuperscript{1}Mirror-symmetric routing assumes that paired nets be implemented using geometrically mirror identical wire segments.
are specified in terms of **absolute** coordinates on a plane. The moves are simple translations (coordinate shifts) or changes in cell orientation – rotations and mirror operations. The cells are allowed to overlap even in illegal ways, as no restriction is made referring to the relative position of a cell with respect to another cell. A penalty cost term – typically, quadratic – is associated with the total illegal overlap, and this penalty must be driven to zero during the minimization of the cost function. The flat representation is well-suited to handle device matching and symmetry constraints – typical to analog layout – since they are easy to model and maintain during successive moves; it also allows to explore the beneficial device overlaps. For these reasons and also its inherent simplicity, the absolute representation was the choice for KOAN/ANAGRAM II [12], PUPPY-A [13], and LAYLA [14] systems.

However, this representation has also shortcomings explained, for instance, in [17]. First, the optimization process is slow: since the exploration space is very large, many moves can yield a small decrease of the cost function. Second, the total illegal overlap (representing only one term of the cost function) is not necessarily equal to zero in the final placement solution: a post-processing step aiming to eliminate the gaps and overlaps must be performed, affecting even more the computation time and degrading the solution optimality. Moreover, the weight of the overlap term in the cost function must be carefully chosen: if it is too large, the search ability of the optimizer for a good placement (in terms of area, total net length, etc.) may be impeded; if it is too small, the cells may have the tendency to collapse since the importance of illegal overlaps is small. To combat this effect, an earlier version of the TimberWolf system [17] used a sophisticated negative control scheme to determine the optimum values of the cost term weights.

The flat representation approach trades off a larger number of moves for easier and quicker to build layout configurations – which may not be always physically realizable though. On the other hand, a second class of placement representations – named **topological** – allows to trade off more complex (but physically correct!) layout constructions per each move of the optimization engine against a smaller number of moves.

### 1.1.6 Topological Representations of the Layout

Different from the flat representation where the cell positions are specified in terms of their coordinates, in a topological representation a placement configuration is **encoded**: the cell positions are relatively specified, based on topological relations between cells. The first popular representations were employing the so-called **slicing model**, introduced by Otten [26]. In this model, the cells are organized in a set of slices, which recursively bisect the layout horizontally and vertically. The direction

---

2 In analog layout, cells can overlap not only in **legal** but also **beneficial** ways ("device merging" or "geometry sharing" [12]).
and nesting of the slices is recorded in a *slicing tree* or, equivalently, in a *normalized Polish expression* [27]. The annealing algorithm (as a typical optimization engine) does not move explicitly the cells – as in the flat representation: the moves are *modifications of the placement codes* (for instance, small reorganizations of a slicing tree, or small changes in a Polish expression that preserve the properties of the encoding). These moves alter *indirectly* the relative positions of the cells. In topological representations, cells cannot overlap illegally, which may lead to an improved efficiency in the placement optimization.

However, the slicing model limits the set of reachable layout topologies. This can degrade layout density, especially when cells are very different in size, which is often the case in analog layout. Furthermore, symmetry and matching constraints are difficult to maintain between successive moves: for instance, a slicing-style placement tool had to implement symmetry constraints in the cost function through the use of *virtual* symmetry axes [28] – a less efficient solution. Although the ILAC system [18] employed slicing trees, it is widely acknowledged today that this model is not a good choice for high-performance analog layout.

After 1995, several novel topological representations, not restricted to slicing floorplan topologies, have been proposed. A remarkably elegant encoding system was proposed by Murata et al., who suggested to encode the “left-right” and “above-below” topological relations using two sequences of cell permutations (see Sect. 1.3 for more details), named a *sequence-pair* [29]. A $O(n^2)$ algorithm ($n$ being the number of cells) based on building a pair of horizontal and vertical constraint graphs was used to construct a compact placement from its encoding, operation called *sequence-pair evaluation*. More recently, a different approach – based on the computation of the longest common subsequence in a pair of weighted sequences – was proposed by Tang et al. [30, 31]. The latest evaluation algorithm achieves a $O(n \log \log n)$ complexity [31] using an efficient model of priority queue [32]. Nakatake et al. devised a meta-grid structure without physical dimensions (called *bounded-sliceline grid* or BSG) to define the topological relations between blocks. The construction of the placement configuration from a BSG is of quadratic complexity [33].

Guo et al. proposed the *ordered tree* (*O-tree*) data structure to reduce the negative effect of code redundancies from the two previous representations [34]. Independently, Chang et al. [35] and Balasa [36] suggested similar representations based on binary trees. These encodings are based on the *natural correspondence* between forests of rooted trees and binary trees [37]. Due to the one-to-one transformation mentioned above, all these tree representations can be regarded as equivalent.

The *corner block list* (CBL) [38] is a representation that is used to encode *mosaic* floorplans (that is, floorplans with zero dead-space). The transitive closure graph (TCG), introduced by Lin and Chang [39], is based on two directed graphs having a node for each cell; their edges correspond to the horizontal and, respectively, vertical topological relations between cells. Different from the tree representations [34, 35], the sequence-pair, the bounded-sliceline grid, the corner block list, and the transitive closure graphs define the topological relations between cells independent of their dimensions.
1.1.7 Selecting a Topological Representation for Analog Placement

The nonslicing topological representations were initially used in block placement and floorplanning tools [40–42]. Could these representations be successfully used in placement tools for analog layout? Which of the topological representations would be better-suited? At a first glance, the main selection criteria should be the same as in block placement: (1) a representation with a low (or even zero) code redundancy to have an exploration space as reduced in size as possible, and (2) the existence of an efficient code evaluation algorithm (preferably of linear complexity) building as fast as possible the placement configuration from the current code in each inner-loop iteration of the simulated annealing.

Without denying the importance of the above criteria, other features specific to analog layout must be taken into account as well. As already explained in Sect. 1.1.4, many analog designs contain an arbitrary number of symmetry groups of devices (that is, groups of devices having distinct symmetry axes), each group containing an arbitrary number of pairs of symmetric devices with the same geometry, as well as self-symmetric devices – presenting a geometrical symmetry and sharing the same axis with its group. Due to this characteristic, most of the codes of any topological representation would be infeasible in symmetry point of view.

In preliminary experiments using sequence-pairs for solving analog placement problems with symmetry constraints [43], a simple exploration scheme was initially attempted: while searching the set of sequence-pairs in a simulated annealing framework, the codes that proved to be infeasible in symmetry point of view during the placement construction were disregarded. Unfortunately, this simple exploration scheme proved to be extremely ineffective, the quality of the placement solutions being very poor. The main reason was revealed to be the huge number of infeasible codes, which were overwhelming in comparison to the “symmetric-feasible” ones.

These preliminary tests showed that symmetry is difficult to model within a topological representation: how to recognize the codes complying with the given set of symmetry constraints without building the corresponding layout? Moreover, assuming the current code is symmetric-feasible, how to prevent the annealer to move from it to an infeasible code? Maintaining the “symmetric-feasibility” of the codes during the annealer’s moves is, in general, a nontrivial task, specific to the topological representation employed: how to restrict the exploration only to the subspace of symmetric-feasible codes?

A topological representation would prove to be a good candidate for solving analog placement problems with symmetry constraints if it possessed a property characterizing codes able to generate placements such that symmetry constraints be satisfied. In the absence of such a property, the fact that a certain representation has an evaluation algorithm of linear complexity is of a lesser importance since most of the codes would be symmetric-infeasible anyway. Such a property would allow to efficiently restrict the exploration to a subspace of “symmetric-feasible” codes.
Several topological exploration techniques for analog placement investigated how to handle symmetry constraints more efficiently. Approaches using sequence-pairs [21, 43], trees [45], and transitive closure graphs [46] have been developed. Notice that the complexity of the code evaluation can be affected when symmetry constraints have to be taken into account. Dealing with an arbitrary number of symmetry groups of devices during the code evaluation necessitates nontrivial algorithmic modifications, paying also a significant computational toll. For instance, the complexity of the evaluation algorithm in [44] is quadratic, while the evaluation algorithm for O-trees in the absence of symmetry constraints is linear [34].

This chapter will present some data structures used in the evaluation of topological representation, followed by a few topological techniques for device-level placement with symmetry constraints.

1.2 Data Structures for Rectilinear Border Contours

This section will give an overview of several data structures that may be used in the evaluation of a given topological representation of the layout. The algorithms in this section are independent of the choice of the topological representation. To emphasize this independence, we shall take into account the horizontal/vertical topological constraints between the cells rather than a certain abstract representation (since the topological constraints are derived from the layout encoding in specific ways that characterize the abstract representation).

1.2.1 Segment Trees

The segment tree, originally introduced by Bentley [48], is a data structure mainly employed in computational geometry, designed to handle operations with intervals whose extremes belong to a given set of coordinates. The coordinates of the intervals can be normalized by replacing each of them by its rank in their minimum-to-maximum order. Therefore, without any loss of generality, we may consider these coordinates as integers in the range \([0, n]\).

The (complete) segment tree is, basically, a rooted binary tree, where each node \(v\) has attached an interval \(v.I = [c, d]\) with integer bounds. If \(d - c > 1\), then node \(v\) has a left and a right descendant – denoted below as \(v.left\) and \(v.right\) – having associated the intervals \([c, \left\lfloor \frac{c+d}{2} \right\rfloor]\) and, respectively, \([\left\lfloor \frac{c+d}{2} \right\rfloor, d]\). The intervals attached to the nodes are called standard, while those pertaining to the leaves and having the length equal to 1 are named elementary.

---

3 Actually, the algorithm in [44] exploits properties of the ordered tree codes which are infeasible in symmetry point of view, to efficiently detect and hence discard them.

4 Very recently, a CAD system for analog layout, including a topological placement tool that uses the corner block list representation [38], has been proposed [47].
A complete segment tree is shown in Fig. 1.2a. This tree is balanced, all the leaves belonging to at most two contiguous levels. The depth of the complete segment tree is $\lfloor \log_2(r - l) \rfloor$, where $[l, r]$ is the interval attached to the root [49].

The segment tree $T(l, r)$ is designed to store intervals whose extremes belong to the set $\{l, l + 1, \ldots, r\}$ in a \textit{dynamic} fashion, that is supporting interval insertions and deletions. The segmentation of an interval $[a, b]$ is completely specified by the operation that stores (inserts) $[a, b]$ in the segment tree $T(l, r)$. To insert an interval, one must visit the nodes in the segment tree along a tour having the following general structure (see Fig. 1.2b): an (possibly empty) initial path $P$ from the root to a node called the \textit{fork} – marked with a star in the figure, from which two (possibly empty) paths $P_L$ and $P_R$ issue. Either the interval being inserted is assigned entirely to the fork (in which case $P_L$ and $P_R$ are both empty), or all the right sons of nodes of $P_L$, as well as all the left sons of nodes of $P_R$ identify the fragmentation of $[a, b]$ into standard intervals. For instance, Fig. 1.2a shows the nodes visited during the insertion of the interval $[4, 6]$ in the segment tree $T(0, 7)$.

The assignment of an interval to a node $v$ of the segment tree could take different forms, depending upon the requirements of the application. Frequently, all we need to know is the cardinality of the set of intervals assigned to any given node $v$. This can be managed by a single nonnegative integer data member $v$.\textit{cnt}, initialized to zero, denoting this cardinality. If this is the case, the assignment of the interval $[a, b]$ to the node $v$ simply becomes $v$.\textit{cnt} = $v$.\textit{cnt} + 1. In other applications, there is need to preserve the identity of the intervals assigned to a node $v$. Then we may append to each node $v$ a secondary data structure, for instance, a singly linked list, whose records are the identifiers of the intervals. Removing an interval from the segment tree works in a symmetric way. Note that only deletions of previously inserted intervals guarantee correctness.

The segment tree is a versatile data structure with numerous applications. It is extremely used especially in the geometric searching algorithms and the geometry of rectangles [49]. For instance, if one wishes to know the number of intervals containing a given point $x$, a simple binary search in the segment tree (that is, the traversal of a path from the root to a leaf) readily solves the problem.
In this section, we are going to use the segment tree data structure to compute the device abscissae $x_i$ assuming the device ordinates $y_i$ are already known \[50\], therefore the extremes of the intervals defining the left and right border contours – the elements of the set $S = \bigcup_i \{y_i, y_i + h_i\}$ – are currently fixed. Also, it is assumed that a topological sort of the horizontal constraint graph is available: this order of visiting the nodes ensures that the blocks to the left are visited before the blocks to the right and, therefore, the horizontal topological constraints would be satisfied.

During the visit of the topologically sorted nodes, a segment tree structure will be gradually built. The creation of the segment tree is done in a top-down manner, starting with the root and expanding the tree till the nodes associated with elementary intervals. In our application, each node $v$ of the segment tree has attached an interval $v.I$ and a value $v.x$ used for the computation of the cell abscissae $x_i$. After each iteration, the segment tree will represent the contour of the right border of the (partial) placement configuration (see the illustrative example towards the end of this section).

First, the $y$-coordinates of the devices are “normalized”: after sorting them increasingly (and eliminating the duplicate values), the $y$-coordinates are replaced by their indexes (ranks) in the ordered sequence. Note that the algorithm operates with intervals $[a_i, b_i]$ rather than $[y_i, y_i + h_i]$, where $a_i, b_i$ are the indices of $y_i$ and, respectively, $y_i + h_i$ in $S$ – the increasingly-sorted set of the interval endpoints. In this way, the size of the segment tree will be kept minimal and, without loss of generality, the $y$-coordinates can hence be considered integers in the range $[0, n]$ ($n$ being the number of devices). Note also that the indices $a_i, b_i$ can be determined while sorting the set $S$ without affecting the complexity of the sorting operation.

**Algorithm: Computation of the device abscissae ($x_i$) using a segment tree**

```
let $x_i = 0$; // reset all the abscissae of the left-bottom corners of the devices

sort increasingly the set $S = \bigcup_i \{y_i, y_i + h_i\}$;
// the duplicate elements of the set are eliminated during sorting

let $m$ be the number of elements of set $S$;
SegmentTreeNode $v_0 = CreateNode ([0, m - 1], 0)$;
// create the root $v_0$ of the segment tree

for each cell $B_i$ (visited in the order of the topological sort)
let $a_i$ be the index of $y_i$, and
let $b_i$ be the index of $y_i + h_i$ in set $S$;
UpdateSegmentTree ($v_0, [a_i, b_i]$);
UpdateRightContour ($v_0, [a_i, b_i]$);
end_for

$W = \max\{v.x\}, \forall v \in \text{SegmentTree}$;
// compute the width $W$ of the placement
```

After the ordinate normalization, the segment tree is recursively built by the procedure $\text{UpdateSegmentTree}$ (see below). The $\text{CreateNode}$ procedure constructs and inserts a new node $v$ in the segment tree – the two parameters being the interval $v.I$ and the value $v.x$. The roots of the left and right subtrees of $v$ are denoted $v.left$ and, respectively, $v.right$; they are initially NULL. The procedure $\text{UpdateSegmentTree}$ is
inserting the normalized interval of \([y_i, y_i + h_i]\) – the spanning of block \(B_i\) along the \(y\) axis – into the segment tree, decomposing it into standard intervals. At the same time, the abscissa \(x_i\) of the left-bottom corner of block \(B_i\) is computed by taking the maximum over all the values \(v.x\) of the nodes with standard intervals.

**procedure UpdateSegmentTree** \((v, [a_i, b_i])\)

- if \(v.I \subseteq [a_i, b_i]\) then
  - if \(v.x > x_i\) then \(x_i = v.x\);
  - else let \(v.I = [c, d]\) and \(mid = \lfloor \frac{c + d}{2} \rfloor\);
    - if \(v\) is currently a leaf of the segment tree then
      - \(v.left = \text{CreateNode}([c, mid], 0)\);
      - \(v.right = \text{CreateNode}([mid, d], 0)\);
    - if \(a_i < mid\) then \(\text{UpdateSegmentTree}(v.left, [a_i, b_i])\);
    - if \(mid < b_i\) then \(\text{UpdateSegmentTree}(v.right, [a_i, b_i])\);

**procedure UpdateRightContour** \((v, [a_i, b_i])\)

- if \(v.I \subseteq [a_i, b_i]\) then \(v.x = x_i + w_i\);
- else let \(v.I = [c, d]\) and \(mid = \lfloor \frac{c + d}{2} \rfloor\);
  - if \(a_i < mid\) then \(\text{UpdateRightContour}(v.left, [a_i, b_i])\);
  - if \(mid < b_i\) then \(\text{UpdateRightContour}(v.right, [a_i, b_i])\);

The computation of each cell abscissa, based on the decomposition of the normalized interval \([y_i, y_i + h_i]\) into standard intervals, is followed by an update of the values \(v.x\) of the visited nodes. To avoid performing any computation twice, the decomposition into standard intervals can be done top-down, using two stacks to store the visited nodes, one for the standard nodes – the “white” nodes in Fig. 1.2b, the other for the nodes on the paths \(P, P_L, \) and \(P_R\) – the “black” nodes in the same figure. Then the update of the values \(v.x\) can be easily done bottom-up. Hence, the implementation of the procedure \(\text{UpdateRightContour}\) can be performed more efficiently than the recursive version given above for reason of clarity.

The decomposition of the root segment into standard intervals is done in \(O(\log n)\) time since the height of the segment tree is at most \([\log_2(m - 1)]\) (the root interval being \([0, m - 1]\)), hence upper-bounded by \([\log_2 n]\). This entails the same complexity for the procedures \(\text{UpdateSegmentTree}\) and \(\text{UpdateRightContour}\). Since the sorting of the set \(S\), together with the computation of the indices \(a_i\) and \(b_i\), take \(O(n \log n)\) time, the overall complexity of the algorithm computing the device abscissae is thus \(O(n \log n)\).

**Example.** Consider a layout with nine rectangular blocks having the widths and heights indicated: \(A(140 \times 30), B(40 \times 20), C(50 \times 50), D(20 \times 60),\)