Chapter 2
State of the Art on Analog Layout Automation

Abstract In the past few years, several tools for the automation of the analog integrated circuit (IC) cell and system layout design, with application on both new and reused designs have emerged. Yet, most of the layout design is still handmade, essentially because analog designers want to have total control over the different design options, and also, due to the fact that current fully automated generators of analog IC layouts produce solutions which are not yet competitive with the manually crafted ones. The state-of-the-art on analog layout automation that follows reveals that after many years of stagnation, electronic design automation (EDA) market is evolving, creating more efficient and complementary approaches to the existing tools. The chapter starts by addressing the placement problem in EDA, providing a brief overview of the most recent placement tools developed, followed by the presentation of the main references of automatic layout generation tools, and the recent advances in layout-aware analog synthesis approaches. Finally, the available commercial solutions for analog layout automation are outlined.

Keywords Analog IC design · Automatic layout generation · Chip floorplan representation · Computer-aided-design · Electronic design automation · Layout-aware synthesis

2.1 Placement

Having the devices for the selected topology sized, they must be laid out in the chip, a common analog layout approach is to split the problem into two smaller problems, placement and routing. An automatic placement tool should produce analog device-level layouts similar in density and performance to the high-quality

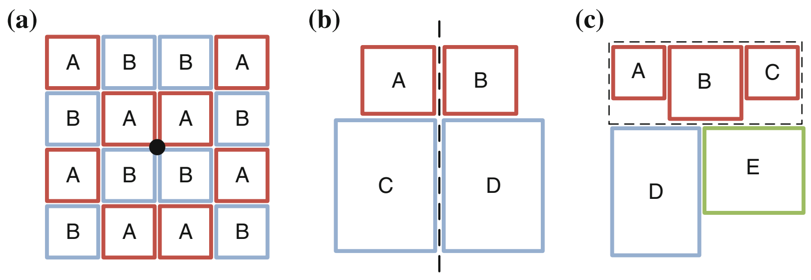

Fig. 2.1 Floorplan constraints **a** Matching (common centroid) **b** Symmetry **c** Proximity (guard ring/well)

manual layouts. In order to achieve this, the capabilities to deal with layout constraints are mandatory.

2.1.1 Layout Constraints

In order to reduce the unwanted impact of parasitic, process variations, different operating conditions and improve the circuit performance, many topological constraints have been introduced into analog placement. The major topological constraints for analog placement are device matching, device symmetry and device proximity [1], as presented in Fig. 2.1. The symmetry constraint 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.

The matching constraint forces a common gate orientation, common centroid or an interdigitized placement among devices, which improves the beneficial effects of the symmetry constraint by reducing the effect of process-induced mismatches. The proximity constraint limits devices to a specific placement so they can share a common substrate/well region, be surrounded by a common guard ring or be placed close to matched devices. Principally, it decreases the effect of substrate coupling, and also avoids large mismatch and deviations during the fabrication process [1, 2].

2.1.2 Chip Floorplan Representations

Each placement tool has its own strategy of representing the cells, two main different approaches to the chip floorplan representation have been used in the last

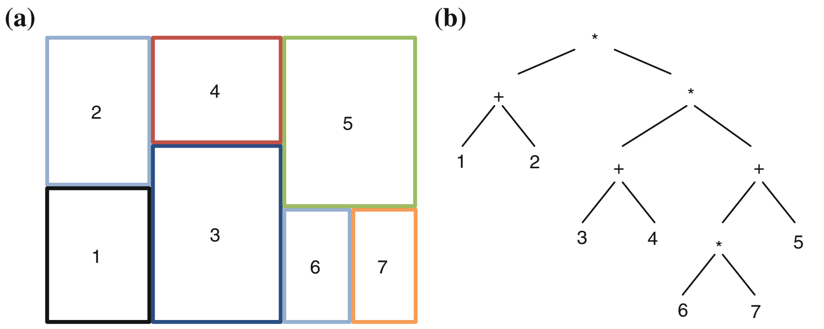

Fig. 2.2 Slicing example **a** Slicing structure, **b** Slicing tree [5]

years [3]: absolute representation (cells are represented by means of absolute coordinates) and topological representation (encoding the positioning relations between any pair of cells), the last one is further classified into slicing or non-slicing representations.

Absolute coordinates is the typical chip floorplan representation used by many computer-aided design (CAD) tools to solve device-level placement problems in analog layout, given the nature of this approach a huge search space is explored. This type of representation allows illegal overlaps during the moves (e.g., translations, changes of orientation), since no restriction is made referring to the relative position of a cell with respect to another cell. To circumvent this situation, a penalty cost term is associated with the total infeasible overlaps, and this penalty must be driven to zero in the, generally, simulated annealing (SA)-based [4] optimization engine [3]. The main disadvantages of using the absolute representation are the high run time, due to the large number of moves necessary to achieve a good layout, once it can generate low-quality and not physically achievable placement solutions, and also, the need of an increased tuning effort due to the difficulty of predicting an appropriate weight for the overlap penalty.

Topological representations trade off a smaller number of moves for more complex-to-build feasible layouts. The first class of topological representation is the slicing model, where cells are organized in sets of slices which 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. The typical simulated annealing-based optimizer does not move cells explicitly, as it does when it operates with absolute layout representation. Instead, it alters the relative positions of cells by modifying the slicing tree or normalized Polish expression encoding the layout [1].

Figure 2.2 represents a slicing structure, which is obtained by recursively cut rectangles into smaller ones, the corresponding oriented rooted binary tree (slicing tree) is also presented. The internal nodes marked with "*" represent vertical cuts, and the ones marked with "+" represent horizontal cuts [5]. Since not all the layout topologies have a slicing structure, this representation can degrade the density of the placement solution, which is more noticeable when the cells of a layout are very different in aspect ratio, a common situation in analog circuits. Also, symmetry and matching constraints have to be implemented in the cost function through the use of virtual symmetry axes, which is a less efficient solution.

The weaknesses identified in the slicing model made it a bad choice for placement tools oriented to high-performance analog layout design, culminated in the emergence of several non-slicing topological representations. For these models, the degradation of layout density is no longer a matter of concern [3, 6]. Within the set of non-slicing structures available nowadays one of the most popular is the sequence pair (SP), which encode the "left–right" and "up-down" positioning relations between cells [7], and the solution space can be effectively explored employing SA or genetic algorithms (GA) [8]. Symmetry and device matching constraints can be easily handled, and has a $O(n^2)$ complexity, where n is the number of placeable cells.

The bounded-sliceline grid (BSG) [9] also has a $O(n^2)$ complexity, it uses a meta-grid structure without physical dimensions, but introduces orthogonal relations of "right-of" and "above" unique for each pair of cells. It poses a more intuitive packing than the sequence pair, although for the SP it is proved the existence of a SP that is unconditionally mapped to an optimal packing, while in the BSG this is not always true and its support of symmetry constraints have not been proved yet.

The ordered tree (O-tree) [10] extended the binary tree to the representation of non-slicing structures, presenting a complexity even smaller than in the slicing floorplan. This method was presented to reduce the drawback of redundancies from SP and BSG representations and also, it needs fewer bits to describe the number of blocks than those methods. The run-time for transforming O-tree to its representing placement is linear to the number of blocks ($O(n)$), so one instance of O-tree will map into exactly one placement, no need for extra computation.

An efficient upgraded representation of binary trees, B^**-tree*, is also available, offers a $O(n.log(n))$ packing for a binary-tree structure that supports cost evaluation, with no need for additional constraint graphs for cost computation, while the other methods above require them [11]. The correspondence between an admissible placement and its induced B*-tree is one-to-one, so no redundancy, with support for symmetry constraints. In Fig. 2.3 an example of a placement with a symmetry constraint is presented, and their respective representation in O-tree and B*-tree encodings.

More recently, Lin et al. [12] introduced the concept of symmetry island, which keeps modules of the same symmetry group connected 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 B*-tree (ASF-B*-tree), which also explores symmetry constraints in two dimensions, unlike the other approaches. 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

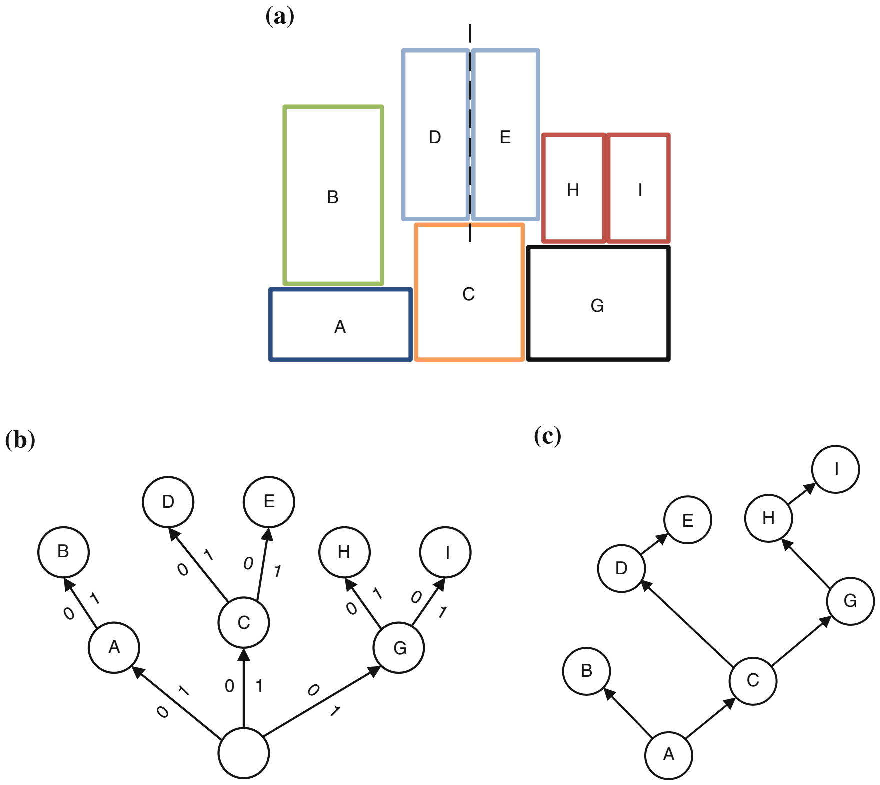

Fig. 2.3 Example of O-tree and B*-tree topological layout representations **a** Placement example **b** O-tree **c** B*-tree [3]

islands and non-symmetry modules, and dynamically update the shape for the devices in a symmetry island. HB*trees are hierarchical oriented, although recent, already proved is high layout quality and runtime efficiency.

The transitive closure graph-based (TCG) [13] representation was proposed to combine the advantages of SP, BSG and B*-tree representations, guaranteeing a unique feasible packing for each representation and that doesn't need to construct additional constraint graphs for the cost evaluation during packing. This approach was quickly replaced for TCG-S [14] derived directly from TCG combined with SP, it presents the fast packing characteristic of SP while maintaining the TCG flexibility to handle placement with special constraints.

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

Table 2.1 Classification of chip floorplan representations

Representation			Advantages	Drawbacks
Absolute			Easier and quicker-to-build layout configurations, every possible placement can be described; Easiness of modeling positioning constraints.	Slow, solution space infinitely large, requiring a high number of moves; Allow illegal overlaps; Solutions not always physically achievable.
Topological	**Slicing**	Slicing Tree / Polish Expr. [5]	Smaller solution space; Moves are modifications of the placement code, changing the relative positions of the cells; Cells cannot overlap illegally, which lead to an improved efficiency.	Symmetry and matching constraints are difficult to maintain; Not all layout topologies have a slicing structure, this representation degrade the density of the solutions.
	Non-slicing		All the advantages of the slicing representations; No degradation of layout density.	
		Sequence Pair [7]	A placement configuration can be derived from any encoding; Handles symmetry and device matching constraints.	*O(n²)* complexity; *O(n.log(n))* extra effort for the computation to construct the placement from a SP.
		BSG [9]	More intuitive packing that the sequence pair.	*O(n²)* complexity; Cannot always represent the optimal packing for a determined group of cells; Deficient support of constraints.
		O-tree [10]	Smaller complexity and less redundancy than SP and BSG, also fewer bits needed to describe the number of blocks; Transforming O-tree to its representing placement is linear, *O(n)* effort.	Less flexible than SP and BSG in representation; Tree structure is irregular, and thus some primitive tree operations (e.g., search, insertion) are not efficient.
		B*-tree [11]	Upgrades the O-tree both in processing as in efficiency; Smaller encoding cost; No need for additional constraint graphs;	Less flexible than SP and BSG in representation.
		HB*-tree [12]	Ability to handle symmetry constraints in 2D with Symmetry Islands and ASF-B*-trees; Possibility to combine Symmetry Islands with the rest of the modules.	Less flexible than SP and BSG in representation.
		TCG-S [14]	Combine the advantages of SP, BSG and B*-tree; Best area utilization, faster convergence speed; Flexibility to handle placement with special constraints.	Despite improvements, the evaluation complexity is still quadratic.

2.1.3 Approaches

Over the years different placement tools have explored the advantages of several chip floorplan representations and new ways of treating layout constraints, these tools had been integrated with more or less success in analog synthesis tools. Next are presented some standalone placement tools, developed recently, that somehow present new solutions or significant developments in the classic placement techniques. A more complete background of analog synthesis tools and respective placement approaches, are referred to the Sect. 2.2 of this chapter.

In the area of device-level placement with layout constraints, Krishnamoorthy et al. [15] presented an algorithm based on the exploration of symmetric-feasible SPs, a symmetry group corresponds to a subset of cells which them all share a common symmetry axis. This approach is powered by a *O(G.n.log(n))* complexity for each code evaluation, where G represents the number of symmetry groups. Koda et al. [16] created an improved method of symmetric placement, obtaining a constraint graph and a set of linear constraint expressions directly from SP while the placement process is accomplished by linear programming.

In the past recent years, developments are being made in hierarchical placement with layout constraints, Lin et al. [17] were the first to present an algorithm for analog placement based on hierarchical module clustering, using HB*-trees. It deals with different constraints simultaneously and hierarchically, this is, if two or more devices are intended to satisfy one or more constraints, they are formed as a cluster, and these clustering constraints can be hierarchically specified to include other clusters. Interesting features for hierarchical symmetry and hierarchical proximity groups that often appear in analog circuits.

In a time dominated by the optimization algorithms, a full deterministic approach arises, Plantage [2], which is based on a hierarchically bounded enumeration of basic building blocks, using B*-trees. This approach is based on the principle that analog circuits show a hierarchical structure, so that hierarchy is used as a bound for the enumeration, aware that a complete enumeration of all possible placements is impracticable. The algorithm begins by generating all placements of the basic modules (leaf nodes of the hierarchy tree), and then the results of the enumerations are combined, guided by the hierarchy tree, until a POF of placements with different aspect ratios for the whole circuit 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.

In a different direction from the other emerging works, Lin et al. [18] proposed a thermal-driven analog placement solution, to simultaneously optimize the placements of “power” and “non-power” (devices which consume much less power than those classified as “power”) devices, in an attempt to annihilate thermally-induced mismatches. It is known that the thermal impact from “power” devices can affect the electrical characteristics of the other thermally-sensitive modules, degrading analog and mixed-signal ICs performance. A thermal profile for a given circuit for better thermal matching of the matched devices is established, and the algorithm evolves until the desired thermal profile is achieved. This thermal profile consists essentially in the even distribution of the heat for the whole chip and requires the temperature at each point in the placement area at each iteration.

2.2 Layout Generation Tools

In this section, some of the milestones in the analog layout generation, along with some recent tools, will be reviewed. In the earliest approaches, procedural module generation techniques coded the entire layout of a circuit in a software tool, 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 graphical user interface (GUI). ALSYN [19] 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 [20]. 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.

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 Intellectual Property Reuse-based Analog IC Layout (IPRAIL) [21] 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].

In order to retain the knowledge of the designer but without forcing an implicit definition, LAYGEN [22] uses a template-based approach to guide the layout generation. ALADIN [23] also allow designers to integrate their knowledge into the synthesis process. While ALG [24] uses the same knowledge-based principle, allowing the designer to interact with the tool in different phases, leaving to the discretion of the designer if the final layout is obtained almost full automatically or by designer directives.

Zhang et al. [25] 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.

The optimization-based layout generation approaches consist of synthesizing the layout solution using optimization techniques according to some cost

Table 2.2 Classification of analog tools based on generation techniques

Procedural		Template	Optimization
Tools	ALSYN [19], Jingnan [20]	IPRAIL [21], LAYGEN [22], ALADIN [23], ALG [24], Zhang [25]	ILAC [26], KOAN/ ANAGRAM II [27], LAYLA [28], Malavasi [29], ALDAC [30]
Advantages	(+) Fast processing; (+) Basic cells	(+) Places modules in a short period of time; (+) Higher abstraction level than procedural; (+) Useful for small adjustments	(+) Higher level of abstraction
Drawbacks	(–) Lack of flexibility, technology migrations force complete cells redesign; high cost of the generation task	(−) Still limits the search space; (−) Designer must add knowledge	(−) Slow; (−) Not always optimal solutions in terms of area and performance

functions, with a higher level of abstraction. The simulated annealing and genetic algorithms are the most common choice for solving analog device-level placement problems, beyond their flexibility in terms of incremental addition of new functionalities; they are relatively easy to implement [6].

In the area of device-level placement with layout constraints there are some main references important to review. ILAC [26] uses simulated annealing operating over a topological slicing tree, used to limit the search space. However, 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/ANALGRAM II [27], LAYLA [28], Malavasi et al. [29] and ALDAC [30]. These methods are usually slow and not always produce optimal solutions in terms of area and 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 of each technique is also highlighted. A summary of the description and functional specifications of the referred tools is presented on Table 2.3, while on Table 2.4 technical specifications and few observations are reported.

2.3 Closing the Gap Between Electrical and Physical Design

In analog IC design, iterations between electrical and physical synthesis to counterbalance layout-induced performance degradations need to be avoided as much as possible [1, 31]. The post-layout performance of a circuit needs to be guaranteed in the presence of layout parasitics, which prevent the circuit from

Table 2.3 Overview of layout generation tools, part I

Layout tool	Years	Specifications			
		Description	M[1]	P[2]	Input/Output
ILAC [26]	1989	Macro-cell Place and Route; placement and routing algorithms inspired by those used for digital design. Not limited to circuits in the input library	✔	✔	In: Netlist, userspecified constraints on cell height; specs. Out: CIF file
KOAN/ ANAGRAM II [27]	1991	Macro-cell Place and Route; uses a pre-defined small module generators data-base. The fusion of two classical tools, placer KOAN and router ANAGRAM	✔	✔	In: Spice netlist with annotation to control place/route; Out: Magic file
ALSYN [19]	1993	Procedural modules controlled though a user-defined database of rules	✔		In: Circuit netlist; rule sets
LAYLA [28] Malavasi [29]	1995 1996	Similar tools. Macro-cell Place and Route. Constraint-driven layout, the degradation of the performance due to due to interconnect parasitic and device mismatches is weighed, combines this with geometrical optimization	✔	✔	In: Circuit netlist; list of performance specifications
Jingan [20]	2001	Automatic generation and reusability of physical layouts; high-functionality pCells	Procedural layout generation		
ALDAC [30]	2002	Generate full-stacked layout modules and performs module placement and local routing. Stacks can be performed either fully-automatically or user controlled	✔		In: Design Rules; ALDAC specific netlist. Out: CIF file
IPRAIL [21]	2003	Automatically creates a template from an existing expertise-embedded layout, and then imposes new device sizes and technology design rules on template	✔		In: CIF file; original and target technology rules. Out: CIF file
LAYGEN [22]	2006	Includes expert knowledge as placement and routing constraints. Designer provides a high level template description and layout is automatically generated	✔		In: Selected template; Technology Design Kit
ALADIN [23]	2006	Designers can develop and maintain technology- and application-independent module generator for relatively complex sub circuits	✔	✔	In: Cells and Netlist; Interative association between them
ALG [24]	2009	User can interact with the tool in each automation step to enhance/polish the layout in order to meet performance specifications. Performance-aware operation provided by a layout adviser tool, YASA	✔	✔	In: Specifications; designer's interaction at different levels
Zhang [25]	2010	Parasitic-aware retargeting; performance sensitivities with respect to parasitics are first determined; automation in a single process without users intervention	✔	✔	In: Existing layout; original and target technology design rules

1—Matching and symmetry constraints; 2—Proximity constraint reaching the expected performance values. This is usually achieved through time-consuming and unsystematic iterations between the electrical and physical design phases. One possible solution involves the integration of these two different phases, by including layout induced effects into the electrical synthesis phase, or sizing at device-level.

This methodology can be found in recent literature with different designations like parasitic-aware, layout-aware and layout-driven synthesis (or sizing). This is a complex and hard to measure process, since there are geometric requirements whose effects on the resulting parasitics are very specific, so they are never included in the traditional electrical synthesis task. If predicted early in the synthesis process, the overestimation of layout-induced parasitics results in wasted power and area, while underestimation may lead to complete malfunction [6]. Knowing the layout induced effects in the synthesis process ensures that performance of the solution is attained after the layout, and that the area minimization is done more realistically.

2.3.1 Layout-Aware Sizing Approaches

Layout-aware synthesis tools target a design process that avoids time consuming iterations, by bringing layout-related data into the sizing process, even while being aware that layout generation, at each iteration, is an expensive process. In Fig. 2.4, a traditional analog design flow with emphasis on circuit sizing is presented, versus the generalized layout-aware methodology proposed in [35]. Next, some state-of-the-art layout-aware synthesis tools and their different ways of extracting layout-parasitics are presented.

Initially, due to the fast processing of basic cells, procedural-based layout generation techniques were used. Vancorenland et al. [32] used manually derived equations along with a procedural layout generation approach to find a suitable solution. Ranjan et al. [33] generates a parameterized layout using the module specification language system, which consists on a fixed template layout, and when the circuit parameters are provided it produces a physical layout. Then, the extracted parasitic from the layout, along with the passive component values are passed to the precompiled symbolic performance models (symbolic equations in terms of circuit parameters), which predicts the circuit performance at each iteration avoiding numerical simulations.

Without actually generate a layout, Pradhan et al. [34] obtains a Pareto-optimal surface with good spread of points for conflicting performance objectives, and each solution contains the specific layout induced effects. The design space is initially sampled to generate circuit matrix models, which predict circuit performances at each iteration. For a uniform random number of design points, layout samples are generated by a procedural layout generator and device parasitics are modeled by linear regression.

Table 2.4 Overview of layout generation tools, part II

Layout tool	Specifications				Observations
	Placement		Router	Development environment	
	Optimization	Floorplan			
ILAC [26]	S. Annealing	Slicing Tree	Best-first maze search	Pascal	(−) Slicing representation
KOAN/ ANAGRAM II [27]	S. Annealing	Absolute	Re-routing, over-the-device wiring, crosstalk avoidance	C code	(−) High dimensionality of the solution space
ALSYN [19]	Deterministic	Slicing Tree	Maze router with crosstalk avoidance	C code	(+) Combines the concept of easy-to-write rules with fast procedural placement
LAYLA [28]	Simulated Annealing	Absolute	Take into account variable wire widths	C++ code	(+) Optimize solution quantifying the performance degradation due to impact of parasitic; more optimum solutions found
Malavasi [29]			Maze router	OCTOOLS	
Jingan [20]	Procedural layout generation			SKILL	Hierarchical parameterized cells
ALDAC [30]	S. Annealing	Absolute	Local routing with two metal layers	C++	Post-layout simulation of multiple layouts
IPRAIL [21]	Linear programming and graph-based methods			–	(−) Undervalues performance
LAYGEN [22]	S. Annealing	B*-tree	Adapts the template routing to the placement	Java	(+) Speeds up retargeting operations
ALADIN [23]	Two-stage: (1) Genetic approach with simulated annealing and half-perimeter routing. (2) Very fast reannealing placement algorithm and global routing			C++, SKILL, Tcl/Tk	(−) Can only handle small or medium size circuits
ALG [24]	Different from custom to automated mode, with global and local routing steps			Java	User may choose the level of automation
Zhang [25]	Mixed-integer nonlinear programming and graph-based methods			C/C++ code	(+) Retargeting with less area and CPU time

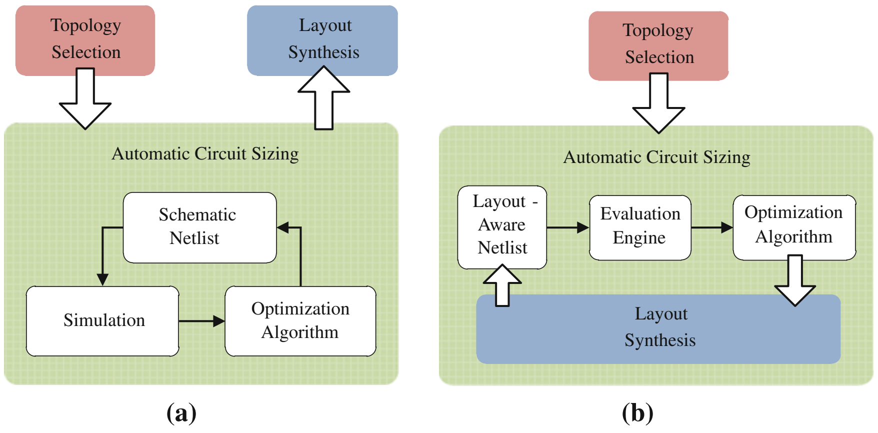

Fig. 2.4 Traditional versus layout-aware circuit sizing flow **a** Traditional analog design flow **b** Layout-aware circuit sizing [35]

Unlike the previous approaches, Castro-Lopez et al. [31] tackles both parasitic-aware and geometrically constrained sizing, which not only includes device parasitic-aware sizing, but also a solution for optimized area and shape. To bypass prohibitively long times for layout generation at each iteration, this approach supports on templates implemented by using the Cadence's pCells technology and SKILL programming. Since the predefined 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 optimal geometric features, or introduced as a constraint to obtain for example, area minimization or a certain aspect ratio. A parasitic estimation without layout generation has been equally implemented, using template sampling techniques and analytical equations.

Recently, Habal et al. [35] ruled out the use of templates given the few degrees of freedom they offer, investigating every possible layout for each device in the circuit using placement algorithm Plantage [2]. The layouts with the best geometric features are kept, and only the final placement selected based on aspect ratio, area and electrical performance is routed. Designer knowledge is supplied by geometric circuit placement and routing constraints, then a deterministic nonlinear optimization algorithm is used for circuit sizing. Table 2.5 shows the summary of the layout aware tools surveyed.

2.4 Commercial Tools

Recently, some commercial solutions have emerged in the analog layout EDA market. Ciranova Helix^{TM} [36] presents as 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 remainder. This perspective is useful on an on-going system-level specifications translation, since the parasitics from the available blocks and estimated areas can be provided for the designer to optimize the circuit. 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. It produces DRC correct placement solutions for design rules from nano-cmos design processes. The output is a standard OpenAccess database that can be edited in most of the layout editors.

In Mentor Graphics IRoute and ARouter [37] 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.

Calibre® YieldAnalyser [37] integrates process variability analysis using model-based algorithms, that automatically plug layout measurements into yield-related equations to identify the areas of the design that have higher sensitivity to process variations. Critical areas can be mathematically weighted by yield impact information to prioritize and trade-off the issues that have the biggest impact on chip yield. YieldAnalyzer performs critical area analysis on all interconnect layers to identify the areas of a layout with excess vulnerability to random particle defects. The tool runs analysis directly on most of the layout data files, e.g., GDSII, OASIS and OpenAccess design databases, and the information is presented in reports and graphs within the designer's layout environment.

Virtuoso® Layout Suite Family [38] 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.

Tanner EDA [39] HiPer DevGen presents a smart generator to accelerate the generation 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. Designers have the control over generation options, layout, 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

Table 2.5 Overview of layout-aware sizing tools

Tool	Years	Estimation of parasitics	LG[1]	Observations
Vancor [32]	2001	Dimensions of the generated layout are used to calculate analytical models	Yes	(−) Performance and information of parasitics is very limited
Ranjan [33]	2004	Device parasitics extracted from the templates are incorporated into symbolic equation performance models	Yes	(−) Limited to small-signal performances; geometric constraints are not considered
Castro-Lopez [31]	2008	Calculation of the MOS diffusions and areas by analytic equations	No	(−) Storage requirements for the lookup table are exceedingly large
		Geometric methods for transistors, 3-D Analytical and Geometric methods for interconnects and other devices	Yes	(+) Extraction very accurate; performance evaluator is HSPICE (−) Longer simulation times
Pradhan [34]	2009	Sample layouts obtained by procedural generation. Parasitics estimated using polynomial models with the known bias, diffusion areas and perimeters	No	(+) Fast; result is a pareto optima156l surface inclusive of layout effects; (−) While device parasitics are approximated; geometric constraints and matching are barely considered
Habal [35]	2011	Parasitic coupling capacitances extracted directly by an integral equation field solver without any modeling or approximation	Yes	(+) SPICE simulations for performance evaluation; (−) Slow, complexity of the sizing method increases with the number of devices, routing and parameters

1—Layout generation at each iteration

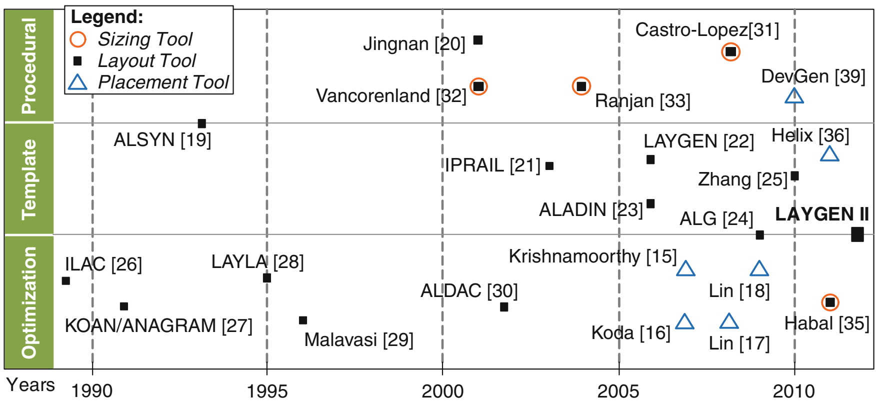

Fig. 2.5 Chronological representation of analog design tools

diodes, etc. For current mirrors there are multiple outputs of different 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 DRC and LVS clean standard cells.

2.5 Conclusions

In this chapter a set of tools applied to analog IC design automation were presented, with emphasis on the layout task. Although much has been accomplished in automatic analog layout generation, the fact is that automatic generators are not yet of standard usage in the industrial design environment. The reviewed approaches are usually limited to some specific circuit or circuit class, and given the large time required to create a new tool or prepare an existing one to support a new circuit, causes analog designers continue to keep on designing the layout manually.

Beyond the efforts made towards the generation of a full 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 the existing layouts, migrating them to new technologies or optimize the old design. Many of the circuits manufactured today are the same ones developed and implemented years ago, so it is extremely important to take advantage of the knowledge embedded in the layout and follow the advances in the integration technologies, instead of going through all the design process again.

At the same time, layout-aware sizing methodologies are spreading and represent an important part of the future of analog design automation, closing the gap between electrical and physical design for a unified synthesis process. Fast, flexible and as complete as possible layout generation techniques are required to include layout-related data into the sizing process, or eventually obtain a final layout simultaneously with the sizing. Most of the layout-aware solutions rely on procedural layout generations, which are known for their difficult reuse and lack of flexibility. The solution of [35] although avoids procedural generations, but at the expense of an increase in the computational time required to complete the automatic flow.

The commercial tools presented, proved that only the approaches that allow for designers to integrate their knowledge into the synthesis process and offer control over the generation, found their way into the EDA market. Most of the available commercial solutions stand out because of the powerful GUIs provided and their characteristics as layout task managers, but lacking on the algorithmic complexity for automatic generation. These tools are used to speed up the manual design task by means of interactive and assisted-edition functionalities.

Figure 2.5 establishes a chronological representation of the tools presented in this chapter, organized by the generation technique used.

From the reviewed approaches, it is possible to notice that floorplan design automation, although far from perfect, is keeping up relatively well with the challenges imposed by new integration technologies. 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.

The idea of parameterized model/template is present in the most recent successful approaches. Increasing the designer's active part in generation can't be seen as a drawback, since the inclusion of his knowledge to guide the process increases the layout quality, and consequently the automatic generation tends to present a satisfying solution for the designer. LAYGEN II also allows the designers to integrate their knowledge into the synthesis process, creating an abstraction layer between technological details and the designer guidelines. This design definition is inherently technology independent, allowing changes in circuit's specifications using the same template, which improves the design reusability and focuses on the efficiency of the retargeting operations.

References

1. H. Graeb, *Analog layout synthesis: a survey of topological approaches* (Springer, Berlin, 2010)
2. M. Strasser, M. Eick, H. Gräb, U. Schlichtmann, F.M. Johannes, Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions in *Proceedings IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 306–313, Nov 2008
3. 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)
4. 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)
5. D.F. Wong, C.L. Liu, A new algorithm for floorplan design, in *Proceedings 23th ACM/IEEE Design Automation Conference (DAC)*, pp. 101–107, Jun 1986

6. 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 and Test in Europe (DATE)*, pp. 274–279, 2009
7. 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(HJul. 2000)
8. A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing* (Springer, Berlin, 2003)
9. 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**, 519–530 (1998)
10. P.-N. Guo, C.-K. Cheng, T. Yoshimura, An O-tree representation of nonslicing floorplan and its applications, in *Proceedings 36th ACM/IEEE Design Automation Conference (DAC)*, pp. 268–273, 1999
11. Y.-C. Chang, Y.-W. Chang, G.-M. Wu, S.-W. Wu, B*-trees: A new representation for nonslicing floorplans, in *Proceedings 37th ACM/IEEE Design Automation Conference (DAC)*, pp. 458–463, 2000
12. P.-H. Lin, S.-C. Lin "Analog placement based on novel symmetry-island formulation, in *Proceedings 44th Design Automation Conference (DAC)*, pp. 465–470, 2007
13. L. Jai-Ming, C. Yao-Wen, TCG: a transitive closure graph-based representation for non-slicing floorplans, in *Proceedings 38th Design Automation Conference (DAC)*, pp. 764–769, 2001
14. 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)
15. K. Krishnamoorthy, S. C. Maruvada, F. Balasa, Topological placement with multiple symmetry groups of devices for analog layout design, in *Proceedings IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2032–2035, May 2007
16. S. Koda, C. Kodama, K. Fujiyoshi, Linear programming-based cell placement with symmetry constraints for analog IC layout. IEEE Trans. Comput. Aided Des. **26**(4), 659–668 (2007)
17. P.-H. Lin, S.-C. Lin, Analog placement based on hierarchical module clustering, in *Proceedings 45th ACM/IEEE Design Automation Conference (DAC)*, pp. 50–55, June 2008
18. P.-H. Lin, H. Zhang, M. Wong, Y.-W. Chang, Thermal-driven analog placement considering device matching, in *Proceedings 46th ACM/IEEE Design Automation Conference (DAC)*, pp. 593–598, Jul 2009
19. V. Meyer, ALSYN: Flexible rule-based layout synthesis for analog ICs. IEEE J. Solid State Circuits **28**(3), 261–268 (1993)
20. X. Jingnan, J. Vital, N. Horta, A SKILLTM—based library for retargetable embedded analog cores, in *Procedings on Design, Automation and Test in Europe (DATE)*, pp. 768–769, Mar 2001
21. 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)
22. N. Lourenço, M. Vianello, J. Guilherme, N. Horta, LAYGEN—Automatic layout generation of analog ics from hierarchical template descriptions, in *Proceedings Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pp. 213–216, Jun 2006
23. 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 (Aug 2006)
24. Y. Yilmaz, G. Dundar, Analog layout generator for CMOS circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **28**(1), 32–45 (2009)
25. L. Zhang, Z. Liu, "A performance-constrained template-based layout retargeting algorithm for analog integrated circuits, in *Proceedings 47th ACM/IEEE Design Automation Conference (DAC)*, pp. 293–298, Jan 2010
26. 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)
27. J.M. Cohn, D.J. Garrod, R.A. Rutenbar, L.R. Carley, KOAN/ANAGRAM II: New tools for device-level analog placement and routing, IEEE J. Solid State Circuits **26**(3) 330–342, Mar 1991
28. 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)

29. 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)
30. P. Khademsameni, M. Syrzycki, A tool for automated analog CMOS layout module generation and placement, in *Proceedings IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 1, pp. 416–421, May 2002
31. 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)
32. P. Vancorenland, G. V. der Plas, M. Steyaert, G. Gielen, W. Sansen, A layout-aware synthesis methodology for RF circuits, in *Proceedings IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 358–362, Nov 2001
33. 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 Design Automation Conference and Test in Europe Conference (DATE)*, vol. 1, pp. 604–609, Feb 2004
34. Pradhan, R. Vemuri, Efficient synthesis of a uniformly spread layout aware Pareto surface for analog circuits, in *Proceedings 22nd International Conference on VLSI Design*, pp. 131–136, Jan 2009
35. 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)
36. "Ciranova," http://www.ciranova.com/
37. Mentor Graphics, http://www.mentor.com
38. Cadence Design Systems Inc, http://www.cadence.com
39. Tanner EDA, http://www.tannereda.com/