3

# Analog and Mixed-Signal Design Strategies

## 3.1 Introduction

The analog designer at the start of the 21st century has several design strategies at his or her disposal to synthesize a well-functioning electronic system. Which strategy to choose depends on issues like the complexity of the system with respect to size and performance requirements, the available computational power, and the technology used to produce the end product. Furthermore, different modeling and synthesis strategies can be selected during different stages of the design process.

This chapter first presents a new classification scheme and brief descriptions of design strategies supported by EDA tools developed by researchers and companies to help the analog designer to select the right approach for the right task. Further, at the end of this chapter, a general new design strategy is introduced and defined to overcome limitations of existing approaches for high-level design of analog and mixed-signal systems.

## 3.2 Classification

The most traditional procedure to design an analog circuit starts with the selection of a topology, usually at circuit level. After analysis with simplified equations, a design plan is formulated to find values for the parameters to obtain correct behavior [103]. Modern design strategies suited for integration in EDA tools differ from this traditional approach in several aspects:

**Place in abstraction–description plane.** Different design strategies may be appropriate at different abstraction and/or description levels. Further, one or more description levels may be included in the synthesis techniques. For example, layout-aware approaches operate on both circuit and physical levels when determining the dimensions of the transistors [117, 98, 44].

**Flexibility.** Some strategies are only applicable to a specific topology or to a class of systems whereas more general approaches can deal with multiple architectures or circuit types. Also, the modeling strategy of the design method may limit its application [32].

**Exploration of architectures.** Several approaches require that the designer selects the entire architecture or parts of it from a limited set of available topologies that the design strategy can deal with [50, 63, 115]. On the other hand, systematic exploration in a design space with different architectures can be part of the synthesis strategy [67, 65, 36].

**Optimization.** To examine the design space, different values for the parameter and/or different architectures must be selected. Whereas a designer can make proposals based on its experience or on analysis, most modern methods for designing analog and mixed-signal systems include an optimization algorithm [56, 46, 11].

These four characteristics of design strategies can be used to distinguish different classes of design approaches. For example, a classification method may focus on the place in the abstraction–description plane (e.g., front-end strategies from specifications to circuit-level description versus back-end strategies from circuit to layout [15]) or on the optimization method used (e.g., circuit sizing methods [46]).

However, gain in circuit performance is not only obtained by finding optimal parameters for a particular architecture, but also by selecting a proper architecture. Indeed, the selected architecture for a specific application can have an inherently worse performance than an alternative one. Therefore, the base criterion for the classification system used in this book is the method to obtain the architecture or topology. Based on this principle, four main classes can be recognized:

1. **Selection before or after dimensioning.** Determining optimal values for the parameters is separated from the selection of the architecture. The designer uses his or her experience or a knowledge-assistant tool to choose a topology. Alternatively, multiple architectures can be dimensioned and afterwards the best solution is selected. Either way, only a limited set of architectures is available in a library. This class is elaborated in Section 3.3.1.
2. **Selection during dimensioning.** The synthesis strategy allows to make an architectural selection during the execution of the algorithm for dimensioning. Topological options are stored in a library as entire architectures or as different choices for subblocks of a generalized architecture. This generalization makes a larger set of architectures available for exploration. Section 3.3.3 summarizes these approaches.
3. **Top-down creation.** Instead of selecting the architecture from a library, the functionality of the system is described at a higher abstraction level, usually with some kind of hardware description language. Via subsequent translation steps, this description is mapped onto a specific topology.

Dimensioning happens either during or after the mapping operation, for example in a constraint transformation step as shown in Section 3.3.3.

4. **Bottom-up generation.** Finally, the architecture can be created starting from a low abstraction level. The design usually starts at the circuit level by connecting individual transistors with each other in a knowledge-based, systematic or stochastic way. It is straightforward to obtain several new circuit topologies with this class of design approaches which is described in Section 3.3.4.

Notice that besides the architecture or topology of a circuit and its nominal parameter values, other important issues should also be considered in analog design, like layout, (substrate) noise effects and robustness regarding technological and operational variations. Although some techniques described in this chapter are also applicable to some of these issues, main emphasis is on methods to obtain the architecture and values for its parameters. Following historic overview illustrates these methods in more detail.

## 3.3 Historic overview

The new classification system based on the method used for architectural exploration is now used to describe the most important design strategies that are supported by CAD tools that have been developed over more than 20 years in the recent past.

### 3.3.1 Selection before or after dimensioning

Traditional design strategies first select the architecture and then alter its parameters to achieve the required performance. If at the end the specifications are not met, costly redesign is needed. The corresponding design flow is depicted in Fig. 3.1. Figure 3.2 shows an example of this design flow for a simple OTA. The proposition of new values by the optimizer corresponds to a transformation applied on the parameters in the generic design flow of Fig. 2.5. Simplification nor refinement operations are present in this simplified design flow. The calculation of the performance happens by evaluation of one of the five types of performance functions listed in Section 2.4.

To fit into the common design habits, a first generation of EDA tools focused on the automation of the two tasks of the traditional design approach: the topology selection and the search for the optimal parameter values. Furthermore, automation of the process to find the parameter values, creates the possibility to select multiple topologies, dimension them and make a final selection afterwards. In case of multi-objective algorithms, multiple combinations of parameter values are returned for a single topology, from which the best one is then selected [29]. Redesign is only necessary when none of the dimensioned topologies meets the specifications. As indicated in Fig. 3.1, such redesign comes down to initially selecting another topology.

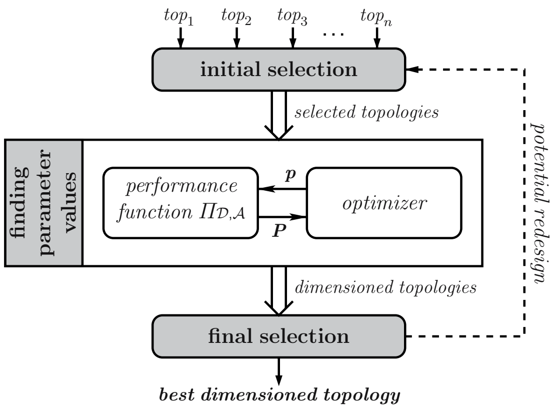


**Fig. 3.1.** Schematic representation of the design strategy applying selection of the topology before or after dimensioning.

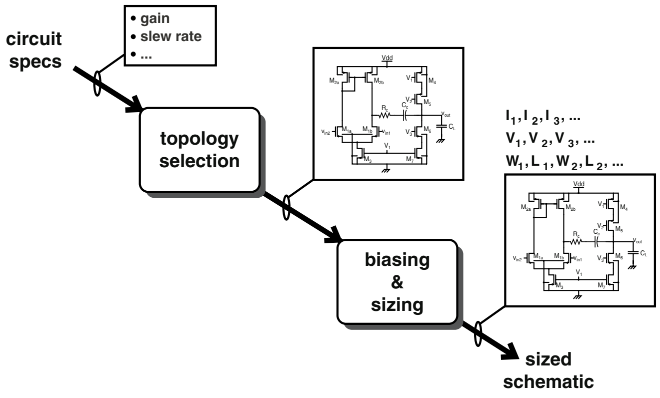


**Fig. 3.2.** Example of a typical design flow for a basic analog cell consisting of topology selection followed by circuit sizing [46].

The design strategy of Fig. 3.1 can be applied on different description levels. For example, for specific building blocks, like op amps, the circuit level is usually selected. Sometimes this is combined with the physical level, e.g., for mixers [117], VCOs [28] or LNAs [87]. On the other hand, the behavioral level is more appropriate for larger systems, like $\Delta\Sigma$ A/D converters [81], PLLs [119] or complete RF systems [22].

## Selection mechanisms

Selecting the appropriate topology for a specific application is a challenging task requiring lots of experience. Therefore, in early design tools (e.g., IDAC

[30]) it is up to the user to make the selection. Other approaches can be divided into three groups:

- *Rule-based selection:* Sets of rules are delivered with each architectural option to guide the selection process.
- *Use of feasibility function:* Comparison of the feasibility function of each topology with the specifications limits the number of topologies to dimension.
- *Trade-off estimation:* The best architecture corresponds to the one with the highest (or lowest) estimated trade-off, expressed as a figure of merit.

Tools have been developed to automate these approaches to a certain extent.

### *Rules*

Each topology available in a library is accompanied by a set of rules indicating when a particular architecture is suited for a specific application. For example, some op amp variant delivers a high gain, but has a small bandwidth. Heuristics provided by an experienced designer form a first base to develop qualitative rules. Further, sizing a circuit for different specifications can be used to automatically derive rules and enhance the knowledge of the selection tool.

For each combination of specifications, the rules should be combined for each available option after which the best circuit is selected. The OPASYN framework [63], for example, stores the knowledge of op amps in a decision tree. At each node a pruning heuristic is added which provides a standard and special decision path for some specification. Finally, one or more leaves remain and are returned as possible topologies. New topologies need to be fitted into the decision tree, which can be a hard task when there are many different performance metrics.

A more systematic approach is adopted in FASY [113]: *fuzzy logic* is used to process and combine the rules for a library of op amps. Membership functions indicating a grade between $[0, 1]$ are defined for variables corresponding to descriptions as ‘small’ or ‘high’. Mathematical manipulation of these variables then results in a total grade for each topology. Once a topology has been selected and sized, a rule can be derived that can become part of the FASY knowledge.

### *Feasibility*

The feasibility function (2.2) provides a straightforward mechanism for architectural selection. Comparison of the specifications with the feasibility space of each topology learns which alternatives can realize the required system [110] and which cannot. The latter are then eliminated.

Instead of deriving the entire feasibility function at once, its use in a selection mechanism can be split up into two steps. A first selection is made by determining the feasible performance intervals without taking into account interdependencies between the different performance specifications. Then, the smaller set of topologies is pruned by calculating the total feasibility function. Consequently, the second, computationally intensive, step should only be applied to a limited number of options (e.g., AMGIE [115]). The feasibility space itself is obtained from symbolic declarative models which describe the performance of different op amp topologies. A major disadvantage is the computational time required which increases rapidly when there are multiple performance characteristics.

*Figures of merit*

Yet another selection mechanism is based on the derivation of a *FoM* for each architecture in the library. In general, a *FoM* is defined as a single characteristic number derived from the implementation properties $P_i$ (e.g., power and area), the required specifications $S_j$ (e.g., gain and bandwidth), and technology characteristics $T_k$ (e.g., minimum length or cut-off frequency):

$$FoM = C \cdot \left[\prod_{i=1}^{n_p} f_i(P_i;\alpha_i)\right] \cdot \left[\prod_{j=1}^{n_s} g_j(S_j;\beta_j)\right] \cdot \left[\prod_{k=1}^{n_t} h_k(T_k;\gamma_k)\right], \tag{3.1}$$

with $C$ a constant and $f_i(\,\cdot\,)$, $g_j(\,\cdot\,)$ and $h_k(\,\cdot\,)$ power functions with $\alpha_i$, $\beta_j$ and $\gamma_h$ as base or exponent. A simple example is the conversion energy for A/D converters [45]:

$$FoM = \frac{1}{2} \cdot P^1 \cdot 2^{-ENOB} \cdot BW^{-1}, \tag{3.2}$$

where no technology parameters are included. Selection of an architecture happens by associating the highest or lowest *FoM* with the best choice.

Designers usually use simplified behavioral models to estimate the *FoM* beforehand (e.g., SDOPT for $\Delta\Sigma$ modulators [80]). Alternatively, a set of already realized designs can be used to derive the parameters in (3.1) [122]. The usefulness of the *FoM* as selection mechanism diminishes if the architectures differ considerably due to the different dependencies of the implementation properties on the specifications and technology parameters.

## Optimization methods

Once one or more architectures have been selected, values for the parameters must be chosen in order to meet the specifications and simultaneously optimize properties like power and area. Several methods have been developed to cope with this optimization problem which typically has a high complexity. Indeed, in analog and mixed-signal systems, many objectives and constraints have to be taken into account. At circuit level, for example, extra constraints about the operating region of the transistors should be included. Interaction between blocks at higher abstraction levels also results in extra constraints. Furthermore, complex device models can make the feasibility space of an analog system quite fanciful. More specifically, the problem is nonlinear and non-convex and multiple local optimal points are encountered. Finally, lack of analytical expressions for the performance function results in the need for time-consuming function evaluations leading to long optimization processes.

Different methods for optimizing the parameters of analog and mixed-signal systems have several properties that distinguish them:

**Single- or multi-objective.** Combining all objectives into one global objective function results in a single-objective approach [11]. On the other hand, multi-objective methods return several solutions from which the best in a certain situation is selected based on the relative importance of the different objectives [127].

**Deterministic or stochastic.** Most optimization algorithms are iterative: they alter temporary solutions by application of parameter steps. Two ways exist to determine new values for parameters during the algorithm. First, a deterministic function is systematically applied on previous values to calculate a step [41]. Each run of the algorithm with the same initial conditions results in the same solution. Alternatively, the new parameter values can be selected based on stochastic variables which may result in different solutions after each run [107].

**Single-point or population.** At each stage of the optimization process, one or more intermediate solutions are stored in a population. Tasks like performance function evaluation for all members of the population can easily be distributed over multiple processors. Further, a new parameter set is derived by taking a step from one point regardless of other intermediate points, or by combining several points to decide about new parameter values.

**With or without derivatives.** To obtain new sets of parameter values, the values of the performance function in the current points are typically required. Some approaches, however, also need the derivatives of this function which results in more complicated calculations at each stage, especially when no analytical expression is available.

**Global or local optimum.** An optimization algorithm may guarantee that the result of the optimization process is the global optimum whereas others can get stuck in a local optimum [95] dependent on the initial point. In practice, the first category returns with high probability a starting point in the neighborhood of the global optimum due to the limited run time.

**Restrictions on modeling strategy.** The optimization algorithm may impose restrictions on the models used to represent the system. For example, continuous functions are usually necessary if derivatives are used. To find a global optimum, a convex optimization problem may be required [32, 73].

**Use of design knowledge.** The optimization of analog and mixed-signal systems can be regarded as a standard mathematical problem without exploiting the knowledge of experienced designers about the system, or include heuristics to find optimal parameter values [105]. Further, some heuristics can be translated into a set of mathematical sizing rules, e.g., to guarantee the functionality or robustness of subblocks [48].

**With or without memory.** Memory can be added to an optimization to simplify the calculation of the performance function or to improve the selection of new sets of parameter values. For example, the already evaluated points and their results are remembered. This memory can be present explicitly [116, 53] or implicitly, e.g., by the constitution of the population. Also, expert tools can be built which store knowledge of previously designed systems.

**Continuous or discrete optimization.** The algorithm selects either values for the parameters from either continuous or discrete sets of values. When continuous values are used, they may be discretized once the final solution has been found (e.g., for transistor dimensions), although this operation can deteriorate the performance.

**Convergence.** The convergence to a final solution may be theoretically guaranteed by an algorithm. On the other hand, the practical convergence rate can be rather slow or the certainty to reach a solution may only come true after an infinite number of iterations.

**Speed.** Finally, the time needed to run by an optimization algorithm depends on most of the previous properties, like the convergence rate, the population size and the complexity of the calculations at each step (e.g., calculating derivatives slows down the process). Further, the execution time can increase drastically with the complexity of the system, e.g., the number of design variables and objectives.

All these properties can result in a different realization which leads to a profusion of optimization approaches. Roughly speaking, six base categories are frequently encountered in CAD tools for analog synthesis published in the literature:

- *Application of knowledge-based design plans, heuristics or rules*
- *Deterministic unconstrained algorithms used for local optimization*
- *Constrained optimization methods solving linear and nonlinear programs*
- *Greedy stochastic optimization techniques*
- *Simulated annealing approaches*
- *Evolutionary computation*

Table 3.1 lists the properties of the base categories according to their elementary implementation in CAD tools. However, practical implementations can be based on a combination of these fundamental methods, e.g., simulated annealing followed by a traditional unconstrained algorithm [113], or a genetic algorithm + simulated annealing method [126].

*Design knowledge*

An analog designer develops a design plan to find the values of the parameters [103]. Such a plan describes how and in which order to calculate all unknown

**Table 3.1.** Overview of properties of base categories of optimization approaches.

| | **Single-/ multi-objective** | **Stoch./ deter-ministic** | **Popula-tion** | **Deriv-atives** | **Global optimum** | **Models** | **Knowl-edge** | **Memory** | **Cont./ discrete** | **Conver-gence** | **Speed** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Design knowledge** | Single-objective | Deter-ministic | Single point | May be included | No guarantee on optimum | Usually analytical models | Heuristics, rules and design plans | In expert systems | Usually continu-ous | If no iterations are used | Fast design plans, but slow if interactive |
| **Local uncon-strained** | Single- and multi-objective | Deter-ministic | Single point | May be included | Only local optimum | Continuous-time models | Only to select initial point | No memory | Usually continu-ous | For well-behaving objective functions | Fast for small-sized problems |
| **Constrained** | Single- and multi-objective | Deter-ministic | Single point | For nonlinear programs | Global for convex programs | Continuous or convex models | For setting up models | No memory | Sometimes discrete mixed with cont. values | Usually conver-gence | Fast for small-sized problems |
| **Greedy stochastic** | Usually single-objective | Stochastic | Sometimes with multiple points | Usually no deriva-tives needed | No guarantee on optimum | No re-strictions | To guide random search | To guide random search | Both con-tinuous and discrete | No guarantee on conver-gence | Slow for large design spaces |
| **Annealing** | Usually single-objective | Stochastic | Basic method is single-point | Usually no deriva-tives needed | Close to global after many iterations | No re-strictions | To select initial point | Only tem-perature in memory | Both con-tinuous and discrete | Probably after enough iterations | Slow for large design spaces |
| **Evolution** | Single- and multi-objective | Stochastic | Population of individ-uals | No deriva-tives needed | Likely close to global after many iterations | No re-strictions | No specific design knowledge | Implicit via popu-lation | Basic GAs use discrete values | Probably after enough iterations | Very slow for large design spaces |

variables defined in the circuit (e.g., transistor widths and lengths) or system (e.g., noise figures and gain factors in an RF transceiver).

The design knowledge should be translated for further use in CAD tools. A straightforward approach is to compile complete design plans for each topology (e.g., IDAC [30]). Multiple plans are needed in the library since different plans are required not only for different topologies, but also for different objectives. A more general system is obtained by building an expert system which also contains general rules and is frequently interactive, e.g., PROSAIC [12], OP–1 [105] and ASIMOV [91]. Nevertheless, the number of architectures that can be dimensioned is limited due to restrictions on the rules' application area. For physical description levels, general placement and routing algorithms can be implemented in a tool which generate automatically an interconnected scheme of standard cells for a wide range of circuits (e.g., CALMOS [9]).

Note that several EDA tools usually contain implicitly design knowledge, for example boundaries and typical values for parameters, initial rough sizing based on simplified models (e.g., STAIC [52], FPAD [40]), or module generators or templates for layouts (e.g., CYCLONE [28]). Explicit incorporation of heuristics as mathematical relations is achieved using the *sizing rules method* [48].

### *Local unconstrained optimizers*

The dimensioning problem is easily translated into a standard mathematical unconstrained optimization problem by defining a scalar function which should be minimized:

$$G(\boldsymbol{p}) = \underbrace{\sum_{k=1}^{N} w_k \cdot f_k(P_k(\boldsymbol{p}))}_{\substack{\text{implementation}\\ \text{properties}}} + \underbrace{\sum_{l=1}^{M} v_l \cdot g_l(P_{l+N}(\boldsymbol{p}), S_l)}_{\substack{\text{performance}\\ \text{specifications}}} + \underbrace{\sum_{m=1}^{C} u_m \cdot h_m(\boldsymbol{p})}_{\substack{\text{structural and sizing}\\ \text{constraints}}}, \quad (3.3)$$

with $\boldsymbol{p}$, $P$ and $S$ the design parameters, performance values and specifications of the system, respectively, and $w_k$, $v_l$ and $u_m$ weight factors. $f_k(\cdot)$, $g_l(\cdot)$ and $h_m(\cdot)$ are deterministic evaluation or penalty functions which can take various forms [93, 86, 89, 40] (e.g., different forms for equality and inequality constraints). For implementation properties, like power and area, identity functions can be used, whereas for specifications usually a shift and normalization is applied. At low abstraction levels, models and specifications for manufacturing or operating tolerances can be taken into account (e.g. [5]). Function (3.3) may be referred to as the *objective*, *error*, *performance* or *cost function.*

Numerous algorithms have been developed to find the minimal value of (3.3) [41]. Several CAD tools contain an implementation of such an algorithm at different levels of complexity: for example, simplex methods (e.g. [86], [56]), gradient-based methods (e.g., OPASYN [63], [86]), several Newton-like approaches (e.g. OAC [93], [11]), and trust region methods (e.g., OPDIC [6]).

A basic condition for correct convergence of these methods is a good starting point. This is derived, for example, by exploiting design knowledge [93]. Another problem is the rapid increase of the execution time with the orders of the parameter and performance spaces, especially when derivatives are needed. Furthermore, analog design problems are usually characterized by various constraints on the parameters. Therefore, constrained optimization methods are more often encountered.

*Constrained optimization*

Instead of making a combination of all objectives and constraints into one function as in (3.3), the problem of finding dimensions can also be considered as a constrained optimization problem, usually written in a standard form:

$$\begin{aligned} \min_{\boldsymbol{p} \in \boldsymbol{p}(\mathcal{D})} \quad & f(\boldsymbol{p}) \\ \text{subject to } & g_k(\boldsymbol{p}) \leq 1 \text{ for } k = 1, \ldots, N \\ & h_l(\boldsymbol{p}) = 1 \text{ for } l = 1, \ldots, M \\ & \boldsymbol{p} > \boldsymbol{0} \end{aligned} \tag{3.4}$$

where the parts of (3.3) with specifications and constraints are explicitly written down as constraints. Depending on the character of the functions in (3.4), the problem is known as a *linear*, *quadratic* or *nonlinear program* [41]. In multi-objective formulations, $f(\boldsymbol{p})$ is replaced by $\boldsymbol{f}(\boldsymbol{p})$ [109].

Since linear programs are only encountered in some special cases (e.g., for filter optimization [100] or layout retargeting [10]), the main focus of analog CAD tools is on nonlinear programs. Several algorithms have been successfully applied for analog designs, like gradient-based approaches (e.g., JiffyTune [21]), methods of feasible directions (e.g., DELIGHT.SPICE [89], [106]), and SQP for op amps [76], analog filters [24], LNAs [87], and general analog cells including statistical analysis (e.g., iEDISON2 [33], WiCkeD [4]).

Like the optimizers listed above, most algorithms can get stuck in a local optimum. However, if all functions in (3.4) are convex, a local solution of (3.4) is also the global optimum. This situation is achieved by using a Geometric Program (GP) which can be transformed into a convex program. In a GP, all equality constraints $h_l(\boldsymbol{p})$ are monomials:

$$h_l(\boldsymbol{p}) = c_l\, p_1^{\alpha_1} \cdots p_D^{\alpha_D}, \quad c_l \in \mathbb{R}^+ \text{ and } \alpha_1, \ldots, \alpha_D \in \mathbb{R}, \tag{3.5}$$

and all functions $f(\boldsymbol{p})$ and $\boldsymbol{g}_k(\boldsymbol{p})$ must be sums of monomials or posynomials. Geometric programming has been applied to the sizing of several analog systems, like CMOS op amps (GPCAD [32], [73]), pipelined A/D converters ([31]), PLLs ([20]), multi-stage amplifiers ([25]), on-chip inductors ([58]), LNAs (ROAD [124]) and systems with statistical variations (e.g., OPERA [125]). The biggest disadvantage of the method is the limitation on the modeling strategy, especially if a model has to be derived for each topology by hand. Possible solutions consist in fitting the performance function with a posynomial model [23] or solving a series of GPs [118]. Of course, this slows down the procedure, making the method less attractive.

*Greedy stochastic optimization*

Greedy algorithms will only accept a new set of parameters if there is some improvement. Elementary stochastic optimization methods like random grid search and random step [56] are not feasible for finding optimal parameter values for an analog system due to the rather extended design space. Instead, they are usually encountered in combination with other approaches.

A straightforward combination is with design knowledge, like heuristics (e.g., iMAVERICK [59]) or information that is derived during the execution of the optimization algorithm [82]. Also, the gradient used in a deterministic approach can be replaced by an estimation based on random perturbations resulting in a stochastic approximation approach (e.g., TOY [111] for yield optimization). A *globally* optimal point is more likely found (but not guaranteed) if random pattern searches are combined with a population-based approach (e.g., ANACONDA [95]).

*Annealing*

Annealing methods mimic aspects of annealing of materials, and are characterized by a cooling mechanism, represented by a decrease of the temperature $T$ by using, for example, a linear or exponential decaying function [60]. Starting from some point $\boldsymbol{p}$ in the parameter space, a new set of parameters $\boldsymbol{p}'$ is derived, by selecting statistically a new point in the neighborhood of the old one [47], or by applying a step of a local optimizer [90]. This new point is accepted if

$$G(\boldsymbol{p}') < G(\boldsymbol{p}) \quad \vee \quad \exp\left(-\frac{\|G(\boldsymbol{p}') - G(\boldsymbol{p})\|}{\mathrm{k}T}\right) > X, \tag{3.6}$$

with $G(\boldsymbol{p})$ an objective function similar to (3.3), $X$ a random number uniformly distributed on $[0, 1]$ and k a normalization constant (Boltzmann's constant if $G(\boldsymbol{p})$ represents energy). As a result, in contrast to the greedy algorithms, up-hill moves have a certain probability to be accepted in annealing approaches and hence the method can escape from local minima. Consequently, the global optimum is theoretically reached after an infinite number of iterations. In a practical implementation, however, it is likely – although not guaranteed – that after enough iterations, the solution is close to the global optimum.

The ability to calculate optimal dimensions without the need for derivatives makes the basic simulated annealing algorithm an attractive choice for optimization in analog CAD tools. Applications include sizing of general analog cells (e.g., OPTIMAN [47], FRIDGE [79], ASTRX/OBLX [90]), op amps (e.g., FASY [113], GBOPCAD [57]), VCOs (e.g., CYCLONE [28]),

$\Delta\Sigma$ modulators (e.g. [102]) and RF receivers (e.g., ORCA [22]), as well as layout generation (e.g., ILAC [101], KOAN/ANAGRAM [19], LAYLA [68], PUPPY-A [72]). In order to speed up the optimization process, several annealing algorithms are executed concurrently on parallel CPUs and operations like recombination commonly found in a genetic evolution process are used to find new points. Examples of such approaches have been implemented for amplifiers in ASF [66] and for layouts in [126].

*Evolution*

Evolutionary algorithms mimic aspects of the natural biological evolution process to find a global optimal solution. A population of individuals (or *phenotypes*) is created where the parameters of a creature are collected in its genome (or *genotypes*). The latter is represented, for example, by a binary string (in original genetic algorithms), real-valued vectors (in evolution strategies [8]) or trees (in genetic programming [64]). Each individual is assigned a *fitness* value corresponding to (3.3) which can be used for ranking and selection.

During the optimization process, new generations are built up by application of genetic operations. The *selection* operator chooses individuals from the population for breeding. *Mutation* perturbs the genotypes to explore new areas in the design space. Finally, *recombination* or *crossover* creates new children by combining the genotypes of the parents. A practical implementation may use only some operations or employ variants (e.g., differential evolution [96]). Usually, adaptations are made to the original algorithms to guarantee convergence to the global optimum (e.g., by employing an elitist selection mechanism). However, similar to the annealing algorithms, this global optimum results only after an infinite number of generations.

The ability to deal with complex structures while only function values of the performance metrics and cost function without derivatives are needed, makes evolutionary methods suited for a wide range of applications. Analog CAD tools based on evolutionary algorithms have been developed to find the optimal parameters of various systems, like RF building blocks (e.g., gaRFeeld [116]), voltage references (e.g. [37]), ADCs (e.g. [121]), op amps (e.g. [123], [2]) and power amplifiers (e.g., M-DESIGN [97]). Evolutionary algorithms can also be employed for multi-objective optimization, e.g., WATSON [29] for op amps and [18] for RF systems. The fitness then becomes a function of the Pareto-dominance of the individuals.

### 3.3.2 Selection during dimensioning

Instead of calculating the optimal dimensions of a particular architecture, and selecting the best architecture beforehand or afterwards, the selection process and the dimensioning are interweaved. As a result, the available design space is enlarged which makes it possible to obtain a better solution compared to the optimization of a fixed topology, be it at larger computational complexity.

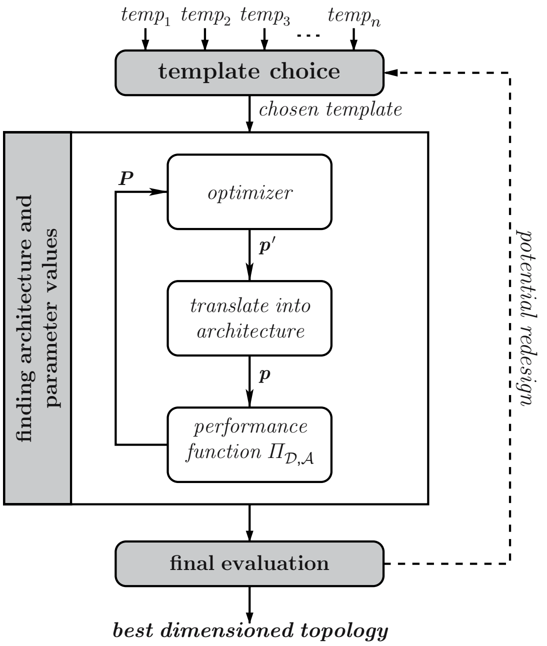

$temp_1$ $temp_2$ $temp_3$ ... $temp_n$
template choice
chosen template
finding architecture and parameter values
$\boldsymbol{P}$
optimizer
$\boldsymbol{p}'$
translate into architecture
$\boldsymbol{p}$
performance function $\Pi_{\mathcal{D},\mathcal{A}}$
potential redesign
final evaluation
best dimensioned topology

**Fig. 3.3.** Schematic representation of the design strategy performing the selection of the topology and the dimensioning concurrently.

Figure 3.3 shows a diagram of the design strategy which combines selection and dimensioning. Similar to the methodology of Fig. 3.1, both approaches are examples of flat design flows summarized in Table 2.3. But rather than selecting a specific architecture, a *template* is chosen explicitly or implicitly. The use of a template or target architecture is also often used in approaches for architectural synthesis of digital systems, e.g., the CATHEDRAL systems [27, 16]. An example of a template for the design of various op amp architectures is shown in Fig. 3.4. Such a template defines a topology in terms of blocks for which different alternatives exist. As a result, all architectural choices available in a library fit into the template. So, the total number of different topologies remains limited. This property ensures that only meaningful architectures are obtained but at the same time no new topologies can be explored. If several templates can be chosen in a tool, a step similar to the initial selection in Fig. 3.1 should be performed.

During the actual optimization, a set of parameters $\boldsymbol{p}'$ selected by the optimizer is translated into a particular architecture and its parameters $\boldsymbol{p}$. Since different architectures are represented during the dimensioning process, the number and meaning of the parameters is variable. Then, the performance function is evaluated. A final evaluation reveals whether the chosen template can deliver a good design. If not, another template has to be chosen and optimized.

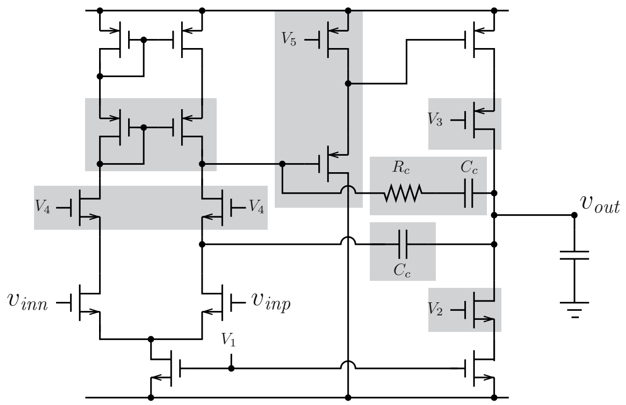


**Fig. 3.4.** Example of a (flat) template with various optional elements indicated by the rectangles, i.e., several optional cascode stages, different compensation schemes and an optional second stage [77].

A key issue in the combination of the selection and dimensioning process, is the definition of the template. Analog CAD tools implementing the design strategy of Fig. 3.3 use different description styles for the templates:

- *Hierarchic:* The template is written as a connection of subblocks for which different types with similar functionality can be chosen during the optimization process.
- *Flat:* All available topologies are characterized by a single description with integer or binary values to indicate different choices.

The specific properties of each group of templates result in different methods to explore the design space and to find an optimal architecture.

## Hierarchical templates

A design strategy based on hierarchical templates has some similarities with a top-down design flow where at each level a separated selection and optimization process is performed. First, general properties common to the different types of a particular subblock are derived. Depending on their actual values, a type is then selected and a specific architecture is generated. Alternatively, a depth-first search algorithm is applied by subsequently selecting all different types. Afterwards, type-specific parameters of the subblocks are determined.

The template, however, imposes several restrictions like the kind of subblocks, their interconnection and the limited number of different types for them. Consequently, the designer has only a rather small number of architectures at his or her disposal.

From the optimization methods summarized in Section 3.3.1, hierarchical templates are most easily fit in approaches based on the application of design knowledge. Indeed, in order to use hierarchical templates, the optimization method should deal with the ability to vary the meaning and number of the parameters in order to represent different architectures. Specific heuristics, rules or design plans are associated with subblocks or their types. An advantage of this approach is that the same knowledge can be used for different templates with common subblocks. Sometimes, a template contains conditional building blocks which can be removed (e.g., an optional second stage).

Several prototype EDA tools implementing a knowledge-based optimization method with hierarchical templates have been developed. Some examples are CAMP [43] which contains different options for *some* subblocks in an op amp, BLADES [39] for three-stage op amp structures, OASYS [50] and ISAID [114] for various op amp architectures. Usually, the end solution of such a tool is further optimized with a local unconstrained optimizer to find the best set of parameters for the architecture obtained by the expert system. Furthermore, during the dimensioning, other optimization approaches can be adopted to find values for the parameters of a specific architecture. For example, SEAS [88] uses simulated annealing for parameter optimization and an expert system to decide which types of subblocks to replace during the optimization.

A complete optimization with evolutionary algorithms of both parameters and architecture represented by flexible hierarchical templates requires the definition of the appropriate operators (i.e., mutation and crossover). For example, such templates and operators are defined in MOJITO [78] for op amp topologies. A major advantage is the possibility to represent a heavily larger number of topologies (about 3,500) than would be possible with flat templates, as discussed in the next subsection.

Another approach to deal with hierarchical templates consists in creating the performance curves for different types of building blocks. The parameters of the blocks are their performance values and hence retain their meaning for different types. A parameter set corresponds directly to a dimensioned subblock with possibly a different number and types of internal parameters [38].

### Flat templates

Whereas knowledge-based approaches are suited to work with hierarchical templates, other optimization methods usually struggle with the difficulty of flexible parameters. Therefore, these methods more naturally deal with a flat template where the parameters altered by the optimizer do not change in number nor meaning. The translation into an architecture maps the parameters onto the properties of possibly different topologies. Since diverse optimization algorithms can be adopted during the design process, flat templates are usually preferred to their hierarchical counterparts.

A major challenge in working with flat templates is to find a suitable system representation. A possible solution consists in defining a topology which combines all accessible architectures by making some connections or entire subblocks optional. A binary variable indicates whether the optional part should be included. Consequently, the system is described by a set of equations which contain both real-valued variables (the normal parameters) and integers (the binary numbers). These equations can be considered as a MINLP. They can be solved using a constrained optimization method combined with a combinatorial optimizer (e.g., a branch-and-bound algorithm). Analog CAD tools based on such an approach have been developed, for example, for CMOS op amps [77] (shown in Fig. 3.4) and $\Delta\Sigma$ modulators [55].

An alternative approach to represent different topologies is to use integers to indicate entire architectures and a *fixed* number of other parameters that are processed by the optimizer regardless of their concrete meaning which can differ in the different topologies. This approach is frequently combined with a stochastic optimization method. First applications are repetitive structures, like filters, where the same parameter is easily re-used for multiple stages since the difference between such stages is merely a scaling factor. If represented as a genome in an evolutionary algorithm, only global parameters and the parameters of each *different* kind of stage should be included to make it possible to reconstruct the entire topology. An example of such an approach is DAISY [42] for $\Delta\Sigma$ modulators.

Yet another way to exploit the flexibility offered by genome representations is to indicate the type of subblock to use by an integer together with a bit string which is translated into the actual parameters of the subblock. As a result, for a system with $n$ building blocks, the operations of the optimizer are always applied on lists with $n$ pairs. Such representation and operations are defined, for example, in DARWIN [67] for three-stage op amps. Infeasible combinations of subblock types are detected with a connection matrix.

### 3.3.3 Top-down creation

A major disadvantage of the design strategies based on selection before, during or after dimensioning is the limited number of architectures that are explored. All available topologies are *selected* from a library, either entirely or as a template with a few options for subblocks or interconnections. In contrast to these design strategies, analog EDA tools which *create* the topology offer a wider design range and the possibility to find new architectures.

A diagram of the corresponding design strategy is depicted in Fig. 3.5. The input is usually a language-based description of the functionality required from the system, written down using a Hardware Description Language (HDL) like VHDL-AMS, sometimes annotated with information to guide the architectural generation process [35]. First, the input description is converted into some internal representation, e.g., a data flow graph [70, 1], a symbolic signal flow graph [54] or a state-space description [3].

Then, the actual optimization process happens in two steps. First, one or more architectures are created by mapping the internal representation onto

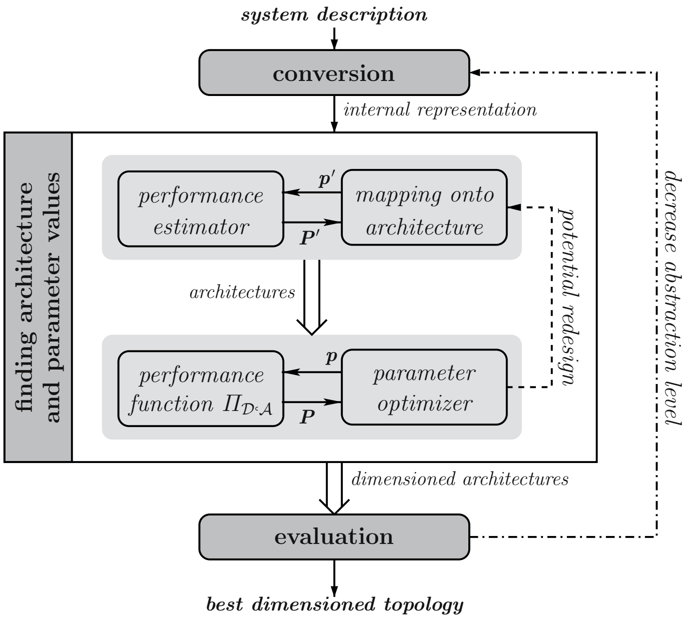
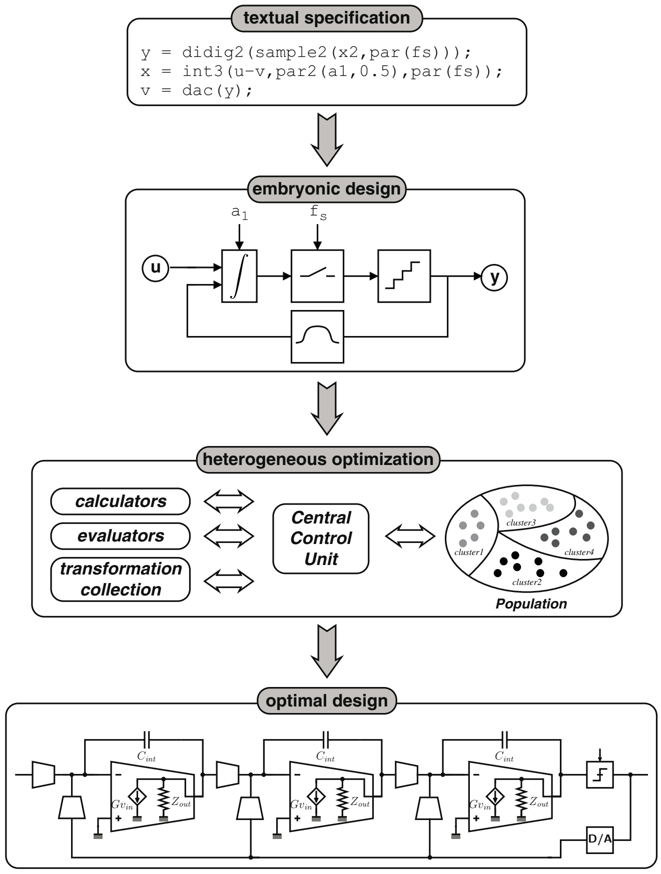

system description
conversion
internal representation
finding architecture and parameter values
performance estimator
$p'$
$P'$
mapping onto architecture
potential redesign
architectures
performance function $\Pi_{\mathcal{D}\cdot\mathcal{A}}$
$p$
$P$
parameter optimizer
decrease abstraction level
dimensioned architectures
evaluation
best dimensioned topology

**Fig. 3.5.** Schematic representation of the design strategy based on top-down creation of the architecture.

a connection of lower-level building blocks. To guide this mapping operation, an estimation of the performance of the architecture can be provided. The second step is the constraints transformation step [17] in which the parameters of the generated architectures are optimized. This step coincides with the application of one of the optimization approaches elaborated in Section 3.3.1. The two steps can also be combined resulting in a simultaneous architectural and parametric optimization process as shown in Fig. 3.6. If no parameters are found to realize the required performance specifications, a redesign is invoked by initiating a new architectural mapping. The dimensioned architectures are evaluated and the optimization is restarted at a lower abstraction level.

## Performance estimation

Estimation of the performance of a particular mapping without determining the values of its parameters is usually difficult and hence inaccurate. As a result, the estimation is often limited to some heuristic values like the complexity of the topology, e.g., the number of building blocks or loops. The algorithm selecting the architectural mapping then uses the results of the performance function via the large feedback loop denoted as ‘potential redesign’ in Fig. 3.5. This performance function is calculated using one of the methods described in Section 2.4.



```
y = didig2(sample2(x2,par(fs)));
x = int3(u-v,par2(a1,0.5),par(fs));
v = dac(y);
```

**Fig. 3.6.** Example of a top-down creation process which starts from a textual description of the required functionality and generates a behavioral model of a specific architecture [75].

For some well-known types of systems, a fitting method can be adopted. The parameters of an analytical expression of the performance function are estimated by applying a fitting algorithm using a database of completed designs. This approach is used, for example, to estimate the power for analog filters in ACTIF [69]. The method is somewhat similar to performance models for calculating the performance functions, but an abstraction is made of the actual implementation. Furthermore, systematically sampling the design space is not possible.

Another approach consists in designing the system down to levels where the performance is more easily estimated, for example using first-order models to speed up the design [34]. Nevertheless, the advantage of using simplified models vanishes if many lower-level choices should be made and the resulting estimation is not accurate enough.

## Architectural mapping mechanisms

The mapping of the internal representation onto an architecture consists of two steps which are usually separated from each other. First, a set of transformations is applied on the internal description which is usually the result of direct translations of all constructs of the language-based system description. For example, if a signal flow graph is used, branches and nodes are eliminated or combined to simplify the model [54].

After the transformation of the internal representation, this description should be mapped onto a particular architecture. Two different types of mappings are used in analog EDA tools:

- *Straight mapping:* A collection of one-to-one relations between the internal representation and the actual topology is available from which the user can select one.
- *Optimized mapping:* Multiple architectures correspond to a specific internal description of the system.

In the first category, the designer performs architectural exploration by providing different input descriptions or selecting another mapping. In the other approach, an optimization algorithm is used to select the architecture in the second approach.

### *Straight mapping*

A traditional variant of a design strategy based on top-down creation uses a block diagram as internal representation. The mapping operations come down to mapping each building block on a lower-level structure [17]. Hence, a collection of one-to-one relations between the internal representation and the actual topology is available. The designer performs architectural exploration by providing different input descriptions.

More flexibility is offered by interactive tools which allow the designer to make some choices during different steps of the straight mapping process, like an implementation style or a transformation. For example, ARCHGEN [3] maps a state-space representation of filters onto different forms of architectures (e.g., observable or controllable) and translates that subsequently into different topologies for which different implementations can be selected (e.g., Gm–C or op amp–C). Another approach employs interactive selection of transformations on admittance matrices to generate filter structures [49].

The interactive character of straight mapping methods makes them suited to quickly examine different options. An experience designer can then decide which architecture should be preferred.

### *Optimized mapping*

Optimized mapping approaches usually use a custom-defined graph type as inner representation. Further, a library of building blocks together with their subgraph representation is defined. Via a pattern recognition algorithm, the graph is then divided into subgraphs corresponding to the blocks in the library. Multiple architectures correspond to the same internal representation. Only the behavior of the blocks at the same abstraction level as the input description should be known for this step.

To find a complete cover of the graph, the search for patterns can be executed in decreasing order of block complexity. Once a pattern is found, the corresponding subgraph is eliminated and this process is repeated until a complete architecture is obtained. The complexity is the only criterion used to direct the optimizer. This approach has been implemented, for example, in TAGUS [54] to explore A/D conversion algorithms.

An alternative heuristic is used in KANDIS [92] for the high-level synthesis of mixed-signal systems. After an interactive partitioning of the internal graph in digital, discrete- and continuous-time subsystems, the last two parts are realized with as few building blocks as possible. Interaction with the user is needed to guide the process if multiple choices are encountered.

The use of performance estimators enables the use of more general optimization algorithms. For example, a branch-and-bound algorithm first generates, based on a set of transformations, all possible filter structures corresponding to a subgraph and then eliminates the topologies for which a large area is estimated [34]. In VASE [36], a tabu-search method is employed to find different architectures for op amp circuits, e.g., to select between blocks operating on voltages or currents. During optimization, the algorithm changes the signal types repetitively to improve the performance and places the performed architectural changes in a tabu list of prohibited options during some iterations. Area is estimated by deriving a first selection of the parameters.

In Antigone [75] (depicted in Fig. 3.6), a straight mapping towards an initial seed is followed by an evolutionary optimization process that generates a complex, more realistic architecture to perform an analog-to-digital conversion. Transformations add or convert architectural elements and parameters or insert implementation details. The required functionality is preserved during the transformations. Hence, this approach combines both types of mapping procedures. This approach is explained in greater detail in Chap. 6.

### 3.3.4 Bottom-up generation

Traditional analog design methodologies analyze a circuit by recognizing fundamental building blocks like a differential pair or a cascode transistor [103]. Extra elements are added to obtain certain effects, e.g., a resistor to cancel a zero, or a capacitor to increase the phase margin. In this way, circuits are created from bottom to top. Therefore, it is a natural choice to support and automate this design strategy by analog EDA tools. More than top-down design approaches, a design methodology starting from low abstraction levels offers opportunities to invent completely new topologies: to map a system's

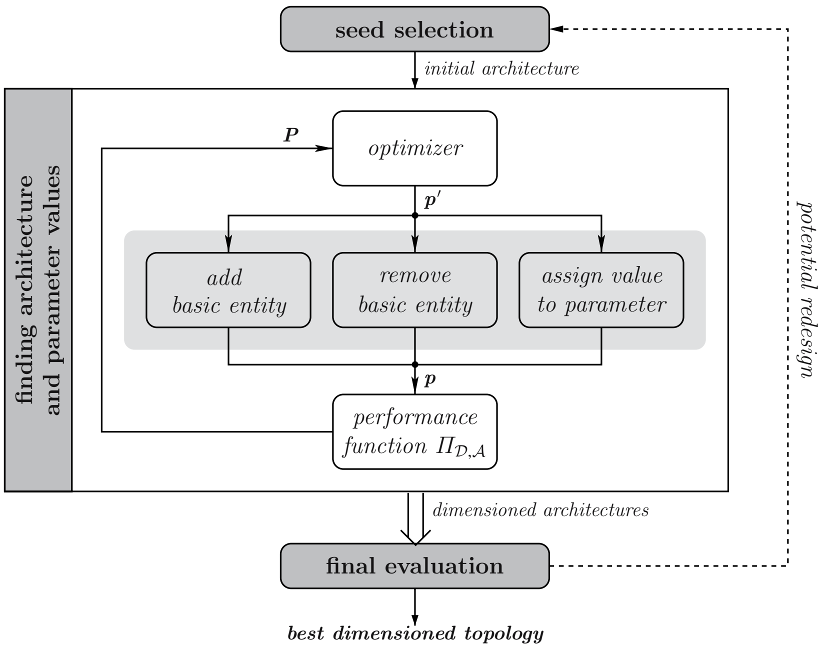
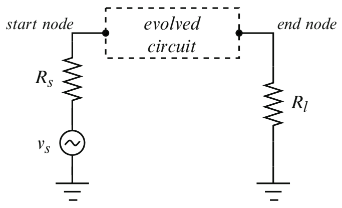

seed selection
initial architecture
finding architecture and parameter values
P
optimizer
p'
add basic entity
remove basic entity
assign value to parameter
p
performance function $\Pi_{\mathcal{D},\mathcal{A}}$
potential redesign
dimensioned architectures
final evaluation
best dimensioned topology

**Fig. 3.7.** Schematic representation of the design strategy based on bottom-up generation of the architecture.

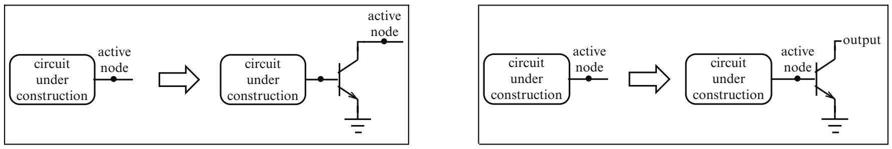
representation onto lower-level building blocks, some assumptions about these blocks should be made, whereas connecting small elementary blocks with each other gives more freedom.

Figure 3.7 shows the outline of the design methodology where the architecture is created in a bottom-up design flow. This design strategy is an example of a simplified generic design flow with properties listed in Table 2.3. The first step implies the selection of an initial architecture or seed which depends on the type and specifications of the system to be designed. This seed should be provided externally by the designer which implies some design experience. During the actual design process, an optimizer selects a transformation: a basic entity is either added or deleted to the architecture, or a value is assigned to a parameter. They can also be combined into one step. Various fundamental entities are possible, like single transistors, elementary building blocks or node connections. An example is given in Fig. 3.8.

Once an architecture has been generated, the performance function is evaluated to provide some clues to the optimizer to make a new selection of transformation. Population-based optimizers will return multiple dimensioned architectures from which the best one is selected. Redesign is only needed if the initial architecture turns out to be ineffective. Since the bottom-up design flow usually starts at circuit level, most information needed to provide a good calculation of the performance function should be available, avoiding redesign.



**(a)** Seed for generating filters


circuit under construction
active node
circuit under construction
active node
circuit under construction
active node
circuit under construction
active node
output

**(b)** Two operations which add an extra element to the circuit

**Fig. 3.8.** Examples of the seed and transformations used to generate analog circuits using a bottom-up design strategy [71].

Applications of the design strategy of Fig. 3.7 differ in both the representation of the system (*modeling strategy*) and the selection of transformations (*synthesis strategy*). Three exploration methods can be distinguished:

- *Knowledge-based:* A designer uses its insight in the working principle of the circuit and determines which elements to add, remove or replace and selects parameter values using standard CAD tools like SPICE and a circuit schematic editor.
- *Exhaustive: All* possibilities with a limited complexity are systematically generated, e.g., as graphs, and evaluated for their benefits.
- *Stochastic:* A stochastic algorithm is adopted to select the operations that will improve the performance values.

For automated synthesis tools, either an exhaustive or stochastic exploration approach should be adopted. The knowledge-based method, on the other hand, resembles the daily practice of analog designers supported mainly by simulation tools.

## Exhaustive exploration

Systematically generating all circuits or systems guarantees that the best architecture is examined. Of course, the computational time increases rapidly with the number of entities in the system. As a result, this method is only practical for pretty small structures.

To create all possibilities, a formal representation for the basic entities is required. For example, a basic element frequently encountered in macro-models for analog circuits is a VCCS. Its operation can be represented as a linear tree-graph with four nodes and a current (output) and a voltage (control) branch [61]. All possible graphs with a limited number of VCCSs are easily generated by defining a number of nodes and drawing all branches between them. After analyzing the resulting circuits, e.g., by calculating the two-port parameters, the useful combinations are selected. Finally, each subgraph is mapped onto a circuit element like a resistor or transistor. Application of this approach results, for instance, in all amplifiers [13] or all CMOS transconductances with maximally two VCCS functions [62]. The bottom-up system generation method is only applied to create an architecture which is symbolically analyzed. The dimensioning process is comparable to the design strategy based on selection of the topology before dimensioning.

## Stochastic exploration

Analog circuits, like op amps, usually contain dozens of transistors [103] despite their relatively simple functionality compared to their digital counterparts. As a result, any random search process in the design space using the circuit description level, tends to be computationally inefficient if no guidance is provided during the operation. A solution is to steer the process by incorporating design knowledge about, for example, basic building blocks and their interconnections as shown in Fig. 3.8. This knowledge, however, can also be derived during the optimization itself, e.g., by analyzing and comparing the generated architectures. Parts from different topologies can then be combined to create new ones. This mechanism naturally leads to evolutionary computation methods.

A major challenge in evolutionary algorithms is the modeling strategy which must be suited to apply the basic genetic operations. For example, recombination of standard circuit-level descriptions is not so trivial. A more appropriate circuit representation technique consists of collecting all selected transformations in Fig. 3.7 in one genome. The result is a program which starts from an *embryonic circuit* and each line inserts a circuit element or changes a connection. Parameters are provided either along with the functions that alter the topology, or with specific commands. To evaluate the performance function, the program is executed and the generated circuit is simulated with a SPICE-like simulator.

The fixed-length programs can directly be used in a genetic algorithm [71]. Crossover comes down to joining two half programs with each other. Alternatively, they can be represented as trees for use with genetic programming techniques [65, 108]. During recombination, subtrees corresponding to subcircuits are exchanged. Note that for these operations, the transformations should be general enough to remain meaningful since they are swapped between programs. Also, the circuit simulator must be able to deal with all generated circuits, which may differ considerably from the commonly used analog circuits. For example, simulation becomes more difficult if there are many (feedback) connections between the nodes. Especially randomly generated circuits used as initial population may cause problems.

### 3.3.5 Overview

A historic overview of most analog EDA tools and methods elaborated above is shown in Fig. 3.9. The vertical axis gives an indication of the abstraction level at which the tool operates. In agreement with Fig. 2.3, low levels correspond to physical (layout) and circuit levels whereas at the top the functional and behavioral level dominate. These description levels are also mentioned in Fig. 3.9.

Early efforts to develop CAD tools were mainly focused on the low levels of abstraction. Whereas tools for layout generation (e.g., CALMOS [9]) can deal with systems containing hundreds or thousands of cells, tools used to determine the sizes of the components like transistors, are usually limited to rather small systems. Large systems are then split into smaller systems by an experienced designer. More recent tools also address more complex systems which are dealt with at high abstraction levels.

The classes corresponding to the applied method to find the topology are denoted by different icons in Fig. 3.9. Following conclusions can be drawn for the four main classes:

**Selection before or after dimensioning.** Optimization of the parameters using one of the methods of Table 3.1 was the first objective of the first-generation design automation methods. The selection of the architecture was the task of the designer. Today, several approaches are well-developed and various analog synthesis tools are commercially available to optimize the parameters of a selected (circuit) topology for analog cells of limited size. Nevertheless, parameter optimization remains a target of research in analog EDA methodologies. Indeed, new technologies, new applications and higher sensitivity to non-ideal phenomena (e.g., deep-submicron effects and substrate noise at continuously higher frequencies) are drivers for research towards tools at all abstraction levels for both small and large systems.

**Selection during dimensioning.** Early methods using templates to select parts of the architecture during the sizing process introduced a rather limited number of architectural options. More recent approaches, however, have shown that a large design space can be explored during the optimization (e.g., more than 3,500 op amp topologies in [78] compared to merely 64 structures in [77]). Hence, at low abstraction levels, these approaches form an alternative for bottom-up generation approaches: various architectures are possible, but the use of a template avoids unknown – and possibly unreliable – architectural results.

**Top-down creation.** Techniques for creating the topology in a top-down flow usually stay at the higher abstraction levels. At lower levels, several

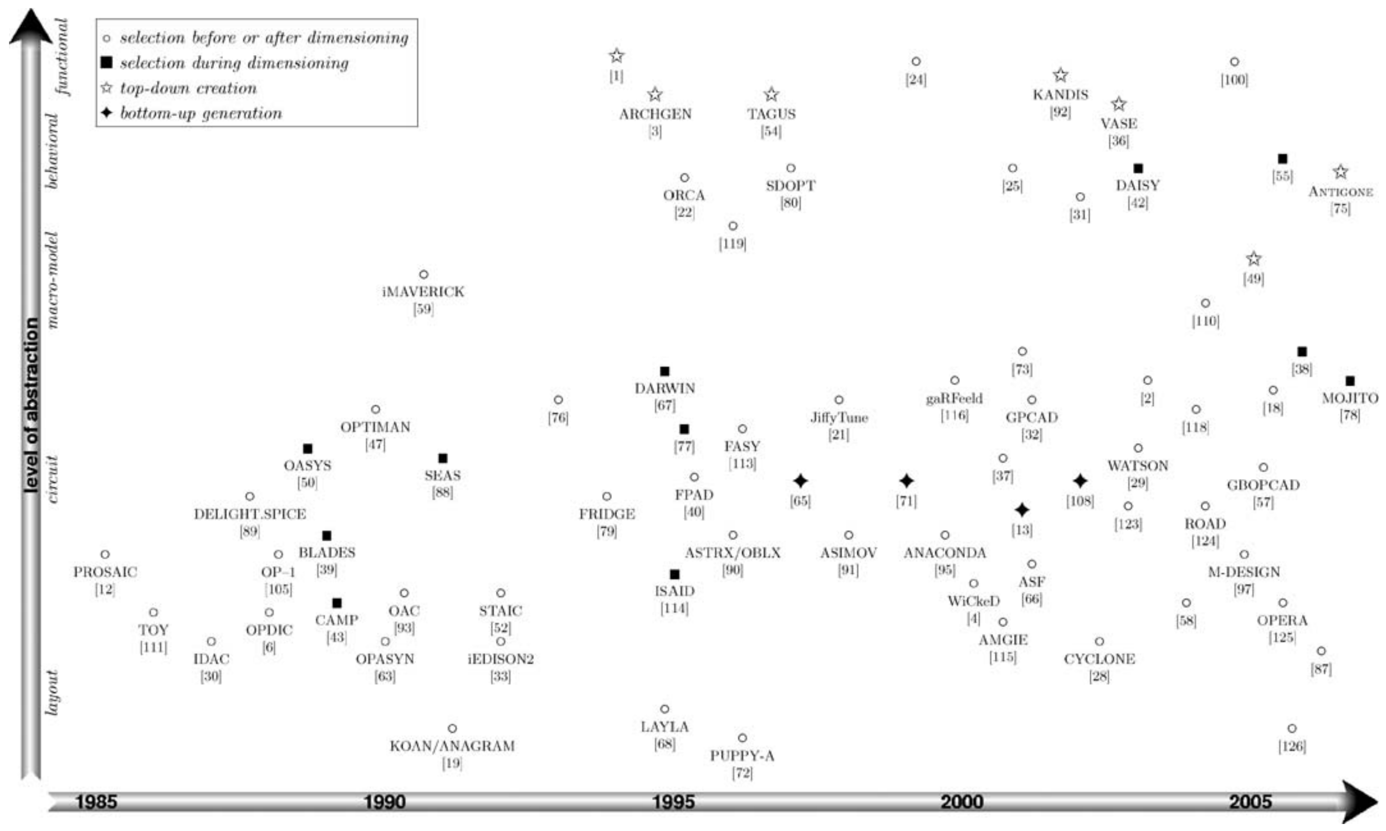


**Fig. 3.9.** Overview of major analog EDA tools for analog synthesis developed in the last 20 years and published in open literature.

effects like the coupling between the building blocks, make it difficult to follow a strict top-down design path. Hence, it may be more appropriate to combine a top-down creation method at behavioral or functional level with the selection of the (circuit) topology before, during or after the sizing process. In the domain of digital synthesis, such approach is known as a 'meet-in-the-middle' strategy [27].

**Bottom-up generation.** Generating the architecture in a bottom-up approach is only achievable with circuit-level descriptions. Two major problems limit the applicability of these methods: the rapid 'explosion' of the design space and corresponding evaluation time, and the possibility to come up with systems which differ too much from well-known analog circuits to be accepted by an analog designer. As a result, most research nowadays focuses on methods of the other three classes.

Approaches to integrate the topology synthesis with the optimization of the parameters, either by selection during dimensioning or via top-down or bottom-up creation, have not made there entrance into commercial analog EDA tools yet. Tools offering automated parameter optimization methods, on the other hand, can be used to design state-of-the-art analog cells or parts of hierarchically decomposed systems. Examples are NeoCircuit of Cadence [14], Circuit Explorer of Synopsis [112], Eldo Optimizer of Mentor Graphics [83], WiCkeD (DesignMD) of MunEDA (ChipMD) [84], and Arsyn of Orora Design Technologies [94]. They all require the selection of the (circuit) topology before optimization of its parameters using one of the methods of Table 3.1. SPICE-like simulations (e.g., with HSpice, Spectre or Eldo) are adopted to evaluate the performance function.

On the other hand, modern VLSI technology makes it possible to integrate complete electronic systems on a single chip (SoC–System-on-Chip) or in a single package (SiP–System-in-Package) [85]. The increased degree of integration results in systems with more functionality, more flexibility, e.g., to cope with different standards, and better performance characteristics, like less power and higher bandwidths. The consequent increase of complexity leads to a growing attention for design of analog and mixed-signal systems in regions of the abstraction–description plane beyond the area of the circuit and physical description levels.

## 3.4 High-level design strategy based on generic behavior

The design strategy developed in this work is based on a projection of the requirements of high-level analog design onto the fundamental properties of design flows elaborated in Chap. 2. The result is a high-level design strategy characterized by the exploitation of generic behavior of analog systems. A formal definition of this strategy is presented in this section.

### 3.4.1 Design flow

Efficient high-level analysis and synthesis pose some requirements on the design strategy:

**Large systems.** As stated in Section 2.2 of Chap. 2, a high-level design strategy in the context of this book is targeted to large complex systems containing analog and mixed-signal electronic circuits. Although the main focus of the developed design strategy is on the analog parts, most systems contain different implementation styles, e.g., in data converters [99] or to improve performance with digital calibration techniques [120]. The modeling and synthesis strategy should deal with these large mixed-signal systems.

**Different abstraction levels.** The performance characteristics of analog systems deteriorate due to physical effects. To analyze their influence during the design, representations of the system at lower abstraction levels are required, even during high-level design. The modeling strategy should support easy addition of extra sources of non-ideal behavior. During synthesis, subsequent refinement operations need to be applied to take these effects into account.

**Different topologies.** Architectural exploration is a key issue in a high-level design methodology. Better results can be expected if there are more topological alternatives available for the synthesis strategy as architectural transformations. Consequently, models for a plethora of different architectures should be easily created. A large design space can be achieved if the modeling strategy allows straightforward representation of newly generated architectures.

**Problems spotting.** Whereas a large design space is beneficial to include the best solution among the available architectures, guidelines are needed to explore it. Apart from the required functionality, the reasons for performance degradation may serve as signposts. To this end, a separation between ideal dominant and non-ideal parasitic signals should be possible with the selected system representation. This segregation allows identification of the problems in the architecture.

**Evaluation.** During synthesis, several architectures with different topological characteristics, parameters and incorporated non-idealities need to be evaluated. A modeling strategy linked to a time-efficient evaluation of the performance function leads to the ability to explore a vast number of options in a reasonable time. Since at high abstraction levels several non-ideal effects are not accounted for, a high-level synthesis strategy should deal with the possibility that the best architecture may be different before and after refinement operations.

**Flexible optimization.** To obtain the best results, the high-level design strategy should be accompanied by a dedicated optimization approach. This algorithm should deal with different architectures with different numbers of parameters and even different performance characteristics (e.g., stability issues are more relevant in feedback than in feedforward structures). Furthermore, an experienced designer will like to give input to guide the optimization process. A flexible optimization strategy takes all these aspects into account.

These requirements can be compared with the general concepts about design flows for analog and mixed-signal systems of Chap. 2 and with the historic overview of the previous section. The following conclusions about an efficient high-level design strategy can be drawn:

**Top-down flow.** To deal with larger systems, the top-down design flow is preferred to the flat and bottom-up methodologies. Furthermore, this approach has the best capabilities for an exploration of system architectures and parameters. However, the top-down paradigm can be combined with bottom-up generated performance models for the building blocks. This results in an approach comparable to the 'meet-in-the-middle' for the design of digital systems [27].

**Behavioral models.** To further simplify the generic top-down design flow, the high-level design is restricted to one description level so that translation operations can be left out. The behavioral description level covers a large part of the abstraction level axis, which makes it straightforward to represent systems at different levels of abstraction. Moreover, ideal and non-ideal signals can be separated more easily in a behavioral model than in a macro-model or circuit description. The actual language used to describe the models depends on the concrete system that has to be designed.

**Generic models.** To easily represent different types of architectures at different abstraction levels, a flexible modeling style is adopted. Instead of a specific behavioral model for a particular system, a generic model is proposed which can be used for a large number of topologies. Via a specialization step, the generic model is then translated into a description for a specific architecture.

**Simulation-based evaluation.** Simulations offer the greatest flexibility for various types of performance values with a limited set-up time for evaluation of the performance function. Therefore, they are the first choice to analyze the behavior of the (non-ideal) system. Computational time is reduced by using simulation algorithms using the specific properties of the generic model. However, for low-level building blocks, the characteristics can be evaluated with performance models [23, 26, 123].

**Evolutionary optimization process.** The high complexity of the optimization problem favors the use of a custom evolutionary method over standard algorithms. Genetic operations can alter both the parameters and the topology which makes it a *heterogeneous* optimization process. Furthermore, with a population-based approach, worse solutions are still investigated so that after introduction of more physical effects, such a solution might turn out to be good. Besides, several solutions can be evaluated in parallel.

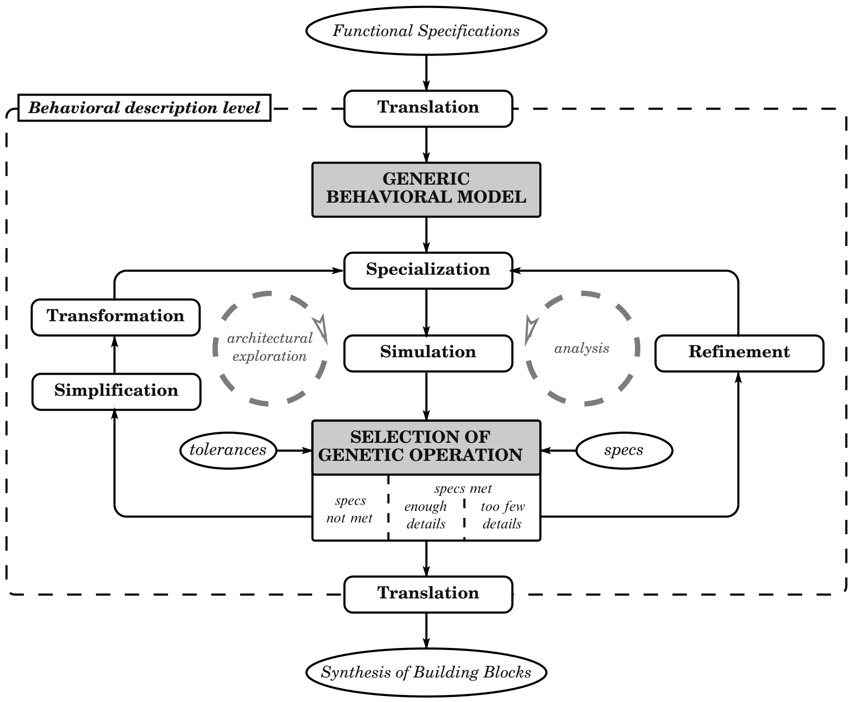


**Fig. 3.10.** High-level design flow for analog and mixed-signal systems based on generic behavior.

A schematic representation of the design flow developed in this work according to these concepts is shown in Fig. 3.10 [74]. In this design flow, the modeling strategy is based on the representation of the system with a *generic behavioral model*. This general description is first derived from the functional specifications. Then, it is specialized to represent a specific architecture at a specific abstraction level. By simulation of this specific behavioral model, the performance values of the system are determined. Based on a comparison of these performance values with specifications and tolerances, the synthesis strategy determines which operations will be applied: a refinement or a transformation of the parameters or the architecture after an optional simplification step. These operations are considered as generalized genetic mutation or recombination steps. Architectural exploration is achieved via subsequent transformations whereas the systematic analysis loop corresponds to the application of refinement operations. This synthesis strategy is a heterogeneous genetic optimization algorithm which will be further elaborated in Chap. 6.

### 3.4.2 Generic behavioral models

A *specific* behavioral model describes the behavior of a particular architecture by a set of mathematical relations between the input and output signals and the states of the system. This concept is generalized by the generic behavioral model defined in this work by following definition [74].

**Definition 3.1 (Generic behavioral model).** *A generic behavioral model of a system describes the functionality in an indirect way as a mathematical interaction between generic functions which represent some relation between signals and states of specific types found in the system.*

To define a generic behavioral model, three kinds of ingredients are needed: alphabets, generic functions and an interaction model.

**Definition 3.2 (Alphabet).** *An alphabet is the smallest set to which a signal or state must belong.*

Defining the alphabets of a system corresponds to specifying the types of all signals and states of the system. Some frequently encountered alphabets in mixed-signal design are:

- $C_0[\mathbb{R}] = \{u | u : \mathbb{R} \rightarrow \mathbb{R} : t \rightarrow u(t)\}$
  ➥ the set of continuous-time signals
- $\mathcal{F}_{C_0[\mathbb{R}]} = \{U | U : \mathbb{R} \rightarrow \mathbb{C} : \omega \rightarrow U(\omega)\}$
  ➥ the set of frequency-domain signals
- $\mathbb{R}^k = \{(x_1, \ldots, x_k) \, | x_i \in \mathbb{R}\}$
  ➥ the set of $k$-dimensional real-valued state vectors
- $\mathbb{B}^a = \{(b_1, \ldots, b_a) \, | b_i \in \{0, 1\}\}$
  ➥ the set of words represented by $a$ binary signals

The formal definition of the input and output alphabets ensures that the model can easily be fitted into a model of a larger system where all building blocks are represented by generic behavioral models. The only condition for connecting an output to an input is a match between the corresponding alphabets.

**Definition 3.3 (Generic function).** *A generic function is a mapping of input elements of particular alphabets onto output elements of particular alphabets described by mathematical parameterized equations.*

Specialization of the generic functions means assigning specific values to the general parameters of the equations. For example, the generic function $\chi$ specifies a down-conversion operation:

$$\chi : C_0[\mathbb{R}] \rightarrow C_0[\mathbb{R}] : u(t) \rightarrow \Re\big\{ \big[u(t) + \mathrm{j}\mathcal{H}\{u(t)\}\big] \cdot \mathrm{e}^{-\mathrm{j}2\pi f_{sh} t} \big\} , \tag{3.7}$$

with $\Re\{a + \mathrm{j}b\} = a$ and $\mathcal{H}\{\,\cdot\,\}$ denotes the Hilbert transform. Specialization implies specifying a value for the frequency shift $f_{sh}$. Another familiar example is a first-order transfer function:

$$h_1 : \mathcal{F}_{C_0[\mathbb{R}]} \rightarrow \mathcal{F}_{C_0[\mathbb{R}]} : U(\omega) \rightarrow K \frac{\mathrm{j}\omega + z}{\mathrm{j}\omega + p} U(\omega) , \tag{3.8}$$

where the gain $K$, pole $p$ and zero $z$ should be specified during specialization. Furthermore, a generic function can contain a variable number of parameters. For example, a filter function may be specified where the number of poles is variable.

**Definition 3.4 (Interaction model).** *An interaction model is an ordered set of instructions that calculate all signals and states of the system via mathematical equations containing generic functions applied on signals and states obtained from either the input or the execution of preceding instructions.*

The interaction model expresses the dynamic behavior of the system in terms of the generic functions. The most straightforward interaction model is just a subsequent application of generic functions $\zeta_i$ on the input signal $x$ and the states $q_i$ belonging to some alphabet:

$$\begin{array}{l} \textbf{input}\, x \\ q_1 \longleftarrow \zeta_1(x) \\ q_2 \longleftarrow \zeta_2(q_1) \\ \vdots \\ q_n \longleftarrow \zeta_n(q_{n-1}) \\ \textbf{output}\, v = q_n \end{array} \tag{3.9}$$

Figure 3.11 depicts a diagram representing a generic behavioral model with this simple interaction model. More complicated interaction models are little programs with loops and conditional branches, for example indicating when signals and states should be calculated at each time point. The time points are then parameters of the interaction model itself. Conditions in the interaction model may result in the elimination of some generic functions from all instructions that are actually executed.

The different elements of the generic behavioral model can be lumped together in a formal machine $M$ which can be represented as an $n$-tuple:

$$M = (\underbrace{A_1, A_2, \ldots}_{\textit{alphabets}}, \underbrace{f_1, f_2, \ldots}_{\substack{\textit{generic}\\ \textit{functions}}}, \underbrace{p_1, p_2, \ldots}_{\substack{\textit{interaction model}\\ \textit{parameters}}}). \tag{3.10}$$

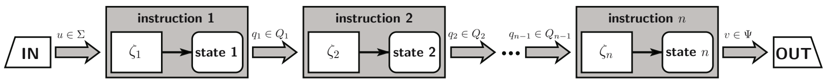


**Fig. 3.11.** Schematic representation of the generic behavioral model with alphabets $\Sigma$, $Q_1, \ldots Q_n$ and $\Psi$, generic functions $\zeta_1, \ldots, \zeta_n$ and interaction model (3.9).

This formal machine describes the generic behavioral model as a model of computation, comparable to commonly used models like a FSM or Turing machine [104]. Execution of the machine's algorithm starts by specialization of the generic functions followed by the application of the interaction model. The representation of a generic behavioral model as a formal machine may explicitly be referred to by the term *formal model*, which should be distinguished from models used for formal verification [51].

The concept of generic behavioral model is a compromise between two traditional approaches: dedicated tools using specific behavioral models (e.g., DAISY [42] for $\Delta\Sigma$ modulators), and the general-purpose simulators like MATLAB/Simulink or simulators of hardware description languages (e.g., a VHDL-AMS simulator [7]). A trade-off is made between the simulation time and the flexibility to represent a wide range of architectures at different abstraction levels, as explained in the next paragraphs.

*Simulation*

Specific behavioral models can be simulated with dedicated algorithms which exploit the specific characteristics of the system. Since only a limited set of architectures and details should be represented, simplifications and speed-up techniques for the simulation can be precompiled in a dedicated tool. As a result, the evaluation of specific behavioral models needs a rather short computational time.

For general-purpose simulators, no system-specific tricks can be applied to make the simulations faster. Compilation of the models written in a general-purpose language is usually offered, but the time needed for this extra step becomes an important delay when several simulations of different architectures with different parameter values are needed in a design flow.

A tool using a generic behavioral model implements the dedicated interaction model. This precompiled simulation algorithm focuses on the properties of the *class* of architectures captured by the generic representation. Different simulation methods can be studied and implemented for different classes. Therefore, it requires less computational time than general-purpose simulators. For example, simulations of the generic behavioral model for $\Delta\Sigma$ modulators developed in this work and presented in Chap. 4, require $6\times$ to $10\times$ less computational time than (time-marching) simulations of a model written in VHDL-AMS. On the other hand, the wider application range comes at the cost of an extra step for the specialization of the generic functions and results in longer simulation times than with specific dedicated tools.

*Flexibility*

Dedicated simulation tools with specific behavioral models allow a user to set some properties like gain, bandwidth or distortion factor. However, these characteristics correspond to a particular architecture represented at some abstraction level. The only way to change the abstraction level is to either select another specific model or to give special values to the parameters to exclude the effect of a non-ideality, like give the value zero to a distortion factor.

With a general-purpose language, all kinds of architectures can be described at any level of detail, which offers a much larger flexibility than usually needed in a design flow. The great freedom for the user, however, has the disadvantage that no inherent structure is present in the models. This lack of structure makes it difficult for a design strategy to systematically manipulate the architecture or its parameters.

Generic behavioral models exhibit a large flexibility within a certain class of systems. Several levels of details can be introduced by providing the appropriate specialization functions. A generic filter function, for example, may be replaced by a model with only dominant poles or by one with also parasitic poles. Although the architecture must fit within the generic template, this template is the result of a translation of the functional specifications so that the set of architectures is limited to the collection of useful systems. The available search space during synthesis can be limited by providing an alternative generic behavioral model. A generic model for A/D converters, for example, can be limited to a specific subtype like $\Delta\Sigma$ converters.

Another benefit of working with a generic template is that there is a link between a part of the model and its contribution to the behavior of the system. For example, a generic function can indicate a frequency conversion, a filtering or a quantization function. Furthermore, ideal and non-ideal behavior can easily be separated by providing corresponding ideal and non-ideal generic functions.

To conclude, EDA tools based on generic behavioral models stand midway between dedicated tools with specific behavioral models and simulators of general-purpose languages. Therefore, they are very suited for a high-level design strategy for analog and mixed-signal systems, and will be used in our design flow.

## 3.5 Conclusions

Several techniques have been developed in research stage over the last twenty years to design analog and mixed-signal systems. The different approaches can be classified based on the treatment of the architecture: it can be selected or created. Selection can occur before, during or after the determination of the values for the parameters. Creation of an architecture happens either via a top-down or a bottom-up design flow. As a result, four major classes of design strategies have been distinguished. Several optimization algorithms exist to select the values of the parameters once the architecture has been chosen. With the aid of a template, several architectural alternatives can be introduced in the optimization. Top-down creation methods perform subsequent mapping operations to translate a description of the functionality into a low-level topology. Finally, bottom-up approaches connect basic entities to 'invent' the architecture.

Based on the concepts of the previous chapter, a top-down high-level design strategy has been introduced based on the representation of the system as generic behavioral models (the *modeling strategy*) and on an evolutionary optimization process (the *synthesis strategy*). This modeling style allows to easily represent a wide range of architectures at different levels of abstraction. The models have a strict structure which makes it straightforward to retrieve the functionality of parts of the model and to identify non-ideal signal flows. Furthermore, dedicated simulation algorithms can be used for time-efficient evaluations. The operations in the abstraction–description plane occurring in the design flow (transformation, refinement and simplification) are translated into the selection of appropriate specialization functions. The heterogeneous optimization algorithm used as synthesis strategy will be described in Chap. 6. The next two chapters of this work elaborate on the development of generic behavioral models in both the time and frequency domain.

## References

[1] A. Achyuthan and M. I. Elmasry. Mixed Analog/Digital Hardware Synthesis of Artificial Neural Networks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1073–1087, Sept. 1994.

[2] G. Alpaydın, S. Balkır, and G. Dündar. An Evolutionary Approach to Automatic Synthesis of High-Performance Analog Integrated Circuits. *IEEE Trans. on Evolutionary Computation*, 7(3):240–252, June 2003.

[3] B. A. A. Antoa and A. J. Brodersen. ARCHGEN: Automated Synthesis of Analog Systems. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 3(2):231–244, June 1995.

[4] K. Antreich, J. Eckmueller, H. Graeb, M. Pronath, F. Schenkel, R. Schwencker, and S. Zizala. WiCkeD: Analog Circuit Synthesis Incorporating Mismatch. In *IEEE Custom Integrated Circuits Conf.*, pages 511–514, Orlando, May 2000.

[5] K. J. Antreich, H. E. Graeb, and C. U. Wieser. Circuit Analysis and Optimization Driven by Worst-Case Distances. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(1):57–71, Jan. 1994.

[6] K. J. Antreich, P. Leibner, and F. Pörnbacher. Nominal Design of Integrated Circuits on Circuit Level by an Interactive Improvement Method. *IEEE Trans. on Circuits and Systems*, 35(12):1501–1511, Dec. 1988.

[7] P. J. Ashenden, G. D. Peterson, and D. A. Teegarden. *The System Designer's Guide to VHDL-AMS.* Morgan Kaufmann Publishers, San Francisco, 2003.

[8] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary Computation: Comments on the History and Current State. *IEEE Trans. on Evolutionary Computation*, 1(1):3–17, Apr. 1997.

[9] H. Beke, W. M. C. Sansen, and R. Van Overstraeten. CALMOS: A Computer-Aided Layout Program for MOS/LSI. *IEEE Journal of Solid-State Circuits*, 12(3):281–282, June 1977.

[10] S. Bhattacharya, N. Jangkrajarng, R. Hartono, and C.-J. R. Shi. Correct-by-Construction Layout-Centric Retargeting of Large Analog Designs. In *IEEE/ACM Design Automation Conf.*, pages 139–144, San Diego, June 2004.

[11] T. Binder, C. Heitzinger, and S. Selberherr. A Study on Global and Local Optimization Techniques for TCAD Analysis Tasks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23(6):814–822, June 2004.

[12] R. J. Bowman and D. J. Lane. A Knowledge-Based System for Analog Integrated Circuit Design. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 210–212, Santa Clara, Nov. 1985.

[13] F. Bruccoleri, E. A. M. Klumperink, and B. Nauta. Generating *All* Two-MOS Transistor Amplifiers Leads to New Wide-Band LNAs. *IEEE Journal of Solid-State Circuits*, 36(7):1032–1040, July 2001.

[14] Cadence Design Systems, Inc. *Virtuoso NeoCircuit Circuit Sizing and Optimization.* 2007. `http://www.cadence.com/datasheets/virneocircuit_ds.pdf`.

[15] L. R. Carley, G. G. E. Gielen, R. A. Rutenbar, and W. M. C. Sansen. Synthesis Tools for Mixed-Signal ICs: Progress on Frontend and Backend Strategies. In *IEEE/ACM Design Automation Conf.*, pages 298–303, Las Vegas, June 1996.

[16] F. Catthoor, J. Rabaey, and H. De Man. Target Architectures in the CATHEDRAL Synthesis Systems: Objectives and Impact. In *IEEE Int. Symp. on Circuits and Systems*, pages 1907–1910, Portland, May 1989.

[17] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli, and I. Vassiliou, editors. *A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits.* Kluwer Academic, 1996.

[18] M. Chu and D. J. Allstot. Elitist Nondominated Sorting Genetic Algorithm Based RF IC Optimizer. *IEEE Trans. on Circuits and Systems—I: Regular Papers*, 52(3):535–545, Mar. 2005.

[19] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley. KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing. *IEEE Journal of Solid-State Circuits*, 26(3):330–342, Mar. 1991.

[20] D. M. Colleran, C. Portmann, A. Hassibi, C. Crusius, S. S. Mohan, S. Boyd, T. H. Lee, and M. del Mar Heshenson. Optimization of Phase-Locked Loop Circuits via Geometric Programming. In *IEEE Custom Integrated Circuits Conf.*, pages 377–380, San Jose, Sept. 2003.

[21] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, C. Visweswariah, and Chai Wah Wu. JiffyTune: Circuit Optimization Using Time-Domain Sensitivities. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1292–1309, Dec. 1998.

[22] J. Crols, S. Donnay, M. Steyaert, and G. Gielen. A High-Level Design and Optimization Tool for Analog RF Receiver Front-Ends. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 550–553, San Jose, Nov. 1995.

[23] W. Daems, G. Gielen, and W. Sansen. Simulation-Based Generation of Posynomial Performance Models for the Sizing of Analog Integrated Circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(5):517–534, May 2003.

[24] N. Damera-Venkata and B. L. Evans. An Automated Framework for Multicriteria Optimization of Analog Filter Designs. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 46(8):981–990, Aug. 1999.

[25] J. L. Dawson, S. P. Boyd, M. del Mar Hershenson, and T. H. Lee. Optimal Allocation of Local Feedback in Multistage Amplifiers via Geometric Programming. *IEEE Trans. on Circuits and Systems—I: Fundamental Theory and Applications*, 48(1):1–11, Jan. 2001.

[26] F. De Bernardinis, M. I. Jordan, and A. Sangiovanni-Vincentelli. Support Vector Machines for Analog Circuit Performance Representation. In *IEEE/ACM Design Automation Conf.*, pages 964–969, Anaheim, June 2003.

[27] H. De Man, J. Rabaey, J. Vanhoof, G. Goossens, P. Six, and L. Claesen. CATHEDRAL-II — a computer-aided synthesis system for digital signal processing VLSI systems. *IEE Computer-Aided Engineering Journal*, 5(2):55–66, Apr. 1988.

[28] C. R. C. De Ranter, G. Van der Plas, M. S. J. Steyaert, G. G. E. Gielen, and W. M. C. Sansen. CYCLONE: Automated Design and Layout of RF LC-Oscillators. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(10):1161–1170, Oct. 2002.

[29] B. De Smedt and G. G. E. Gielen. WATSON: Design Space Boundary Exploration and Model Generation for Analog and RF IC Design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):213–224, Feb. 2003.

[30] M. G. R. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B. L. A. G. Goffart, E. A. Vittoz, S. Cserveny, C. Meixenberger, G. van der Stappen, and H. J. Oguey. IDAC: An Interactive Design Tool for Analog CMOS Circuits. *IEEE Journal of Solid-State Circuits*, 22(6):1106–1116, Dec. 1987.

[31] M. del Mar Hershenson. Design of pipeline analog-to-digital converters via geometric programming. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 317–324, San Jose, Nov. 2002.

[32] M. del Mar Hershenson, S. P. Boyd, and T. H. Lee. Optimal Design of a CMOS Op-Amp via Geometric Programming. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 20(1):1–21, Jan. 2001.

[33] A. Dharchoudhury and S. M. Kang. An Integrated Approach to Realistic Worst-Case Design Optimization of MOS Analog Circuits. In *IEEE/ACM Design Automation Conf.*, pages 704–709, Anaheim, June 1992.

[34] A. Doboli, N. Dhanwada, A. Nunez-Aldana, and R. Vemuri. A Two-Layer Library-Based Approach to Synthesis of Analog Systems from VHDL-AMS Specifications. *ACM Trans. on Design Automation of Electronic Systems*, 9(2):238–271, Apr. 2004.

[35] A. Doboli and R. Vemuri. Behavioral Modeling for High-Level Synthesis of Analog and Mixed-Signal Systems From VHDL-AMS. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(11):1504–1520, Nov. 2003.

[36] A. Doboli and R. Vemuri. Exploration-Based High-Level Synthesis of Linear Analog Systems Operating at Low/Medium Frequencies. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(11):1556–1568, Nov. 2003.

[37] Dongkyung Nam, Yun Deuk Seo, Lae-Jeong Park, Cheol Hoon Park, and Bumsup Kim. Parameter Optimization of an On-Chip Voltage Reference Circuit Using Evolutionary Programming. *IEEE Trans. on Evolutionary Computation*, 5(4):414–421, Aug. 2001.

[38] T. Eeckelaert, R. Schoofs, G. Gielen, M. Steyaert, and W. Sansen. Hierarchical Bottom–up Analog Optimization Methodology Validated by a Delta–Sigma A/D Converter Design for the 802.11a/b/g Standard. In *IEEE/ACM Design Automation Conf.*, pages 25–30, San Francisco, July 2006.

[39] F. El-Turky and E. E. Perry. BLADES: An Artificial Intelligence Approach to Analog Circuit Design. *IEEE Trans. on Computer-Aided Design*, 8(6):680–692, June 1989.

[40] M. Fares and B. Kaminska. FPAD: A Fuzzy Nonlinear Programming Approach to Analog Circuit Design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):785–793, July 1995.

[41] R. Fletcher. *Practical Methods of Optimization.* Wiley, Chichester, 1987.

[42] K. Francken and G. G. E. Gielen. A High-Level Simulation and Synthesis Environment for $\Delta\Sigma$ Modulators. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(8):1049–1061, Aug. 2003.

[43] A. H. Fung, B. W. Lee, and B. J. Sheu. Self-Reconstructing Technique for Expert System-Based Analog IC Designs. *IEEE Trans. on Circuits and Systems*, 36(2):318–321, Feb. 1989.

[44] Gang Zhang, A. Dengi, R. A. Rohrer, R. A. Rutenbar, and L. R. Carley. A Synthesis Flow Toward Fast Parasistic Closure For Radio-Frequency

Integrated Circuits. In *IEEE/ACM Design Automation Conf.*, pages 155–158, San Diego, June 2004.

[45] G. Geelen. A 6b 1.1GSample/s CMOS A/D Converter. In *IEEE Int. Solid-State Circuits Conf.*, pages 128–129, San Francisco, Feb. 2001.

[46] G. G. E. Gielen and R. A. Rutenbar. Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits. *Proceedings of the IEEE*, 88(12):1825–1854, Dec. 2000.

[47] G. G. E. Gielen, H. C. C. Walscharts, and W. M. C. Sansen. Analog Circuit Design Optimization Based on Symbolic Simulation and Simulated Annealing. *IEEE Journal of Solid-State Circuits*, 25(3):707–713, June 1990.

[48] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich. The Sizing Rules Method for Analog Integrated Circuit Design. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 343–349, San Jose, Nov. 2001.

[49] D. G. Haigh, F. Q. Tan, and C. Papavassiliou. Systematic Synthesis of Active-RC Circuit Building-Blocks. *Analog Integrated Circuits and Signal Processing*, 43(3):297–315, June 2005.

[50] R. Harjani, R. A. Rutenbar, and L. R. Carley. OASYS: A Framework for Analog Circuit Synthesis. *IEEE Trans. on Computer-Aided Design*, 8(12):1247–1266, Dec. 1989.

[51] W. Hartong, L. Hedrich, and E. Barke. Model Checking Algorithms for Analog Verification. In *IEEE/ACM Design Automation Conf.*, pages 542–547, New Orleans, June 2002.

[52] J. P. Harvey, M. I. Elmasry, and B. Leung. STAIC: An Interactive Framework for Synthesizing CMOS and BiCMOS Analog Circuits. *IEEE Trans. on Computer-Aided Design*, 11(11):1402–1417, Nov. 1992.

[53] Hongzhou Liu, A. Singhee, R. A. Rutenbar, and L. R. Carley. Remembrance of Circuit Past: Macromodeling by Data Mining in Large Analog Design Spaces. In *IEEE/ACM Design Automation Conf.*, pages 437–442, New Orleans, June 2002.

[54] N. C. Horta and J. E. Franca. Algorithm-Driven Synthesis of Data Conversion Architectures. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16(10):1116–1135, Oct. 1997.

[55] Hua Tang and A. Doboli. High-Level Synthesis of $\Delta\Sigma$ Modulator Topologies Optimized for Complexity, Sensitivity, and Power Consumption. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(3):597–607, Mar. 2006.

[56] L. P. Huelsman. Optimization—A Powerful Tool for Analysis and Design. *IEEE Trans. on Circuits and Systems—I: Fundamental Theory and Applications*, 40(7):431–439, July 1993.

[57] Jie Yuan, N. Farhat, and J. Van der Spiegel. GBOPCAD: A Synthesis Tool for High-Performance Gain-Boosted Opamp Design. *IEEE Trans. on Circuits and Systems—I: Regular Papers*, 52(8):1535–1544, Aug. 2005.

[58] Jintae Kim, Jaeseo Lee, L. Vandenberghe, and Chih-Kong Ken Yang. Techniques for Improving the Accuracy of Geometric-Programming Based Analog Circuit Design Optimization. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 863–870, San Jose, Nov. 2004.

[59] Y.-C. Ju, V. B. Rao, and R. A. Saleh. Consistency Checking and Optimization of Macromodels. *IEEE Trans. on Computer-Aided Design*, 10(8):957–967, Aug. 1991.

[60] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May 1983.

[61] E. A. M. Klumperink, F. Bruccoleri, and B. Nauta. Finding All Elementary Circuits Exploiting Transconductance. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 48(11):1039–1053, Nov. 2001.

[62] E. A. M. Klumperink and B. Nauta. Systematic Comparison of HF CMOS Transconductors. *IEEE Trans. on Circuits and Systems—II: Analog and Digital Signal Processing*, 50(10):728–741, Oct. 2003.

[63] H. Y. Koh, C. H. Séquin, and P. R. Gray. OPASYN: A Compliler for CMOS Operational Amplifiers. *IEEE Trans. on Computer-Aided Design*, 9(2):113–125, Feb. 1990.

[64] J. R. Koza. *Genetic Programming. On the Programming of Computers by Means of Natural Selection.* Bradford Book, Cambridge, 1992.

[65] J. R. Koza, F. H. Bennett, III, D. Andre, M. A. Keane, and F. Dunlap. Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming. *IEEE Trans. on Evolutionary Computation*, 1(2):109–128, July 1997.

[66] M. J. Krasnicki, R. Phelps, J. R. Hellums, M. McClung, R. A. Rutenbar, and L. R. Carley. ASF: A Practical Simulation-Based Methodology for the Synthesis of Custom Analog Circuits. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 350–357, San Jose, Nov. 2001.

[67] W. Kruiskamp and D. Leenaerts. DARWIN: CMOS opamp Synthesis by means of a Genetic Algorithm. In *IEEE/ACM Design Automation Conf.*, pages 433–438, San Francisco, June 1995.

[68] K. Lampaert, G. Gielen, and W. M. Sansen. A Performance-Driven Placement Tool for Analog Integrated Circuits. *IEEE Journal of Solid-State Circuits*, 30(7):773–780, July 1995.

[69] E. Lauwers and G. Gielen. Power Estimation Methods for Analog Circuits for Architectural Exploration of Integrated Systems. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 10(2):155–162, Apr. 2002.

[70] E. A. Lee and T. M. Parks. Dataflow Process Networks. *Proceedings of the IEEE*, 83(5):773–799, May 1995.

[71] J. D. Lohn and S. P. Colombano. A Circuit Representation Technique for Automated Circuit Design. *IEEE Trans. on Evolutionary Computation*, 3(3):205–219, Sept. 1999.

[72] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli. Automation of IC Layout with Analog Constraints. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(8):923–942, Aug. 1996.

[73] P. Mandal and V. Visvanathan. CMOS Op-Amp Sizing Using a Geometric Programming Formulation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 20(1):22–38, Jan. 2001.

[74] E. Martens and G. Gielen. Generic Behavioral Modeling of Analog and Mixed-Signal Systems for Efficient Architectural-Level Exploration. In *ECSI Forum on Specifications and Design Languages*, pages 15–22, Darmstadt, Sept. 2006.

[75] E. Martens and G. Gielen. Top-Down Heterogeneous Synthesis of Analog and Mixed-Signal Systems. In *IEEE/ACM Design, Automation and Test in Europe Conf. and Exhibition*, pages 275–280, Munich, Mar. 2006.

[76] P. C. Maulik, L. R. Carley, and D. J. Allstot. Sizing of Cell-Level Analog Circuits Using Constrained Optimization Techniques. *IEEE Journal of Solid-State Circuits*, 28(3):233–241, Mar. 1993.

[77] P. C. Maulik, L. R. Carley, and R. A. Rutenbar. Integer Programming Based Topology Selection of Cell-Level Analog Circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):401–412, Apr. 1995.

[78] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert. Simultaneous Multi-Topology Multi-Objective Circuit Sizing Across Thousands of Analog Circuit Topologies. In *IEEE/ACM Design Automation Conf.*, pages 944–947, San Diego, June 2007.

[79] F. Medeiro, F. V. Fernández, R. Domínguez-Castro, and A. Rodríguez-Vázquez. A Statistical Optimization-Based Approach for Automated Sizing of Analog Cells. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 594–597, San Jose, Nov. 1994.

[80] F. Medeiro, B. Pérez-Verdú, J. M. de la Rosa, and Á. Rodríguez-Vázquez. Using CAD Tools for Shortening the Design Cycle of High-Performance Sigma–Delta Modulators: A 16.4 bit, 9.6 kHz, 1.71 mW $\Sigma\Delta$M in CMOS 0.7 $\mu$m Technology. *Int. Journal of Circuit Theory and Applications*, 25(5):319–334, Sept. 1997.

[81] F. Medeiro, B. Pérez-Verdú, A. Rodríguez-Vázquez, and J. L. Huertas. A Vertically Integrated Tool for Automated Design of $\Sigma\Delta$ Modulators. *IEEE Journal of Solid-State Circuits*, 30(7):762–772, July 1995.

[82] S. Mehrotra, P. Franzon, and Wentai Liu. Stochastic Optimization Approach to Transistor Sizing for CMOS VLSI Circuits. In *IEEE/ACM Design Automation Conf.*, pages 36–40, San Diego, June 1994.

[83] Mentor Graphics. *Eldo Datasheet.* 2007. `http://www.mentor.com/products/ic_nanometer_design/custom_design_simulation/eldo/upload/eldods.pdf`.

[84] MunEDA. *WiCkeD – Product Overview.* 2007. `http://www.muneda.com/pdf/WiCkeD\%20Product\%20Overview.pdf`.

[85] B. Murari. Bridging the Gap Between the Digital and Real Worlds: the Expanding Role of Analog Interface Technologies. In *IEEE Int. Solid-State Circuits Conf.*, pages 30–35, San Francisco, Feb. 2003.

[86] N. S. Nagaraj. A New Optimizer for Performance Optimization of Analog Integrated Circuits. In *IEEE/ACM Design Automation Conf.*, pages 148–153, Dallas, June 1993.

[87] A. Nieuwoudt, T. Ragheb, and Y. Massoud. SOC-NLNA: Synthesis and Optimization for Fully Integrated Narrow-Band CMOS Low Noise Amplifiers. In *IEEE/ACM Design Automation Conf.*, pages 879–884, San Francisco, July 2006.

[88] Z.-Q. Ning, T. Mouthaan, and H. Wallinga. SEAS: A Simulated Evolution Approach for Analog Circuit Synthesis. In *IEEE Custom Integrated Circuits Conf.*, pages 5.2.1–5.2.4, San Diego, May 1991.

[89] W. Nye, D. C. Riley, A. Sangiovanni-Vincentelli, and A. L. Tits. DELIGHT.SPICE: An Optimization-Based System for the Design of Integrated Circuits. *IEEE Trans. on Computer-Aided Design*, 7(4):501–519, Apr. 1988.

[90] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley. Synthesis of High-Performance Analog Circuits in ASTRX/OBLX. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(3):273–294, Mar. 1996.

[91] I. O'Connor and A. Kaiser. Automated Design of Switched-Current Cells. In *IEEE Custom Integrated Circuits Conf.*, pages 477–480, Santa Clara, May 1998.

[92] P. Oehler, C. Grimm, and K. Waldschmidt. A Methodology for System-Level Synthesis of Mixed-Signal Applications. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 10(6):935–942, Dec. 2002.

[93] H. Onodera, H. Kanbara, and K. Tamaru. Operational-Amplifier Compilation with Performance Optimization. *IEEE Journal of Solid-State Circuits*, 25(2):466–473, Apr. 1990.

[94] Orora Design Technologies, Inc. *Arsyn Circuit Synthesis Platform for Automated Analog, RF and Digital Transistor Circuit Design.* 2007. `http://www.orora.com/Products-arsyn/`.

[95] R. Phelps, M. Krasnicki, R. A. Rutenbar, L. R. Carley, and J. R. Hellums. Anaconda: Simulation-Based Synthesis of Analog Circuits Via Stochastic Pattern Search. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):703–717, June 2000.

[96] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution. A Practical Approach to Global Optimization.* Springer, Berlin, 2005.

[97] J. Ramos, K. Francken, G. G. E. Gielen, and M. S. J. Steyaert. An Efficient, Fully Parasitic-Aware Power Amplifier Design Optimization Tool. *IEEE Trans. on Circuits and Systems—I: Regular Papers*, 52(8):1526–1534, Aug. 2005.

[98] M. Ranjan, W. Verhaegen, A. Agarwal, H. Sampath, R. Vemuri, and G. Gielen. Fast, Layout-Inclusive Analog Circuit Synthesis using

Pre-Compiled Parasitic-Aware Symbolic Performance Models. In *IEEE/ACM Design, Automation and Test in Europe Conf. and Exhibition*, pages 604–609, Paris, Feb. 2004.

[99] B. Razavi. *Principles of Data Conversion System Design.* Wiley, New York, 1995.

[100] J. Ren and M. Greenstreet. A Unified Optimization Framework for Equalization Filter Synthesis. In *IEEE/ACM Design Automation Conf.*, pages 638–643, Anaheim, June 2005.

[101] J. Rijmenants, J. B. Litsios, T. R. Schwarz, and M. G. R. Degrauwe. ILAC: An Automated Layout Tool for Analog CMOS Circuits. *IEEE Journal of Solid-State Circuits*, 24(2):417–425, Apr. 1989.

[102] J. Ruiz-Amaya, J. de la Rosa, F. V. Fernández, F. Medeiro, R. del Río, B. Pérez-Verdú, and A. Rodríguez-Vázquez. High-Level Synthesis of Switched-Capacitor, Switched-Current and Continuous-Time $\Sigma\Delta$ Modulators Using SIMULINK-Based Time-Domain Behavioral Models. *IEEE Trans. on Circuits and Systems—I: Regular Papers*, 52(9):1795–1810, Sept. 2005.

[103] W. M. C. Sansen. *Analog Design Essentials.* Springer, Dordrecht, 2006.

[104] J. E. Savage. *Models of Computation. Exploring the Power of Computing.* Addison-Wesley, Reading, 1998.

[105] B. J. Sheu, A. H. Fung, and Y.-N. Lai. A Knowledge-Based Approach to Analog IC Design. *IEEE Trans. on Circuits and Systems*, 35(2):256–258, Feb. 1988.

[106] J.-M. Shyu, A. Sangiovanni-Vincentelli, J. P. Fishburn, and A. E. Dunlop. Optimization-Based Transistor Sizing. *IEEE Journal of Solid-State Circuits*, 23(2):400–409, Apr. 1988.

[107] J. C. Spall. *Introduction to Stochastic Search and Optimization. Estimation, Simulation, and Control.* Wiley, Hoboken, 2003.

[108] T. Sripramong and C. Toumazou. The Invention of CMOS Amplifiers Using Genetic Programming and Current-Flow Analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(11):1237–1252, Nov. 2002.

[109] G. Stehr, H. Graeb, and K. Antreich. Performance Trade-off Analysis of Analog Circuits By Normal-Boundary Intersection. In *IEEE/ACM Design Automation Conf.*, pages 958–963, Anaheim, CA, June 2003.

[110] G. Stehr, H. Graeb, and K. Antreich. Analog Performance Space Exploration by Fourier-Motzkin Elimination with Application to Hierarchical Sizing. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 847–854, San Jose, CA, Nov. 2004.

[111] M. A. Styblinski and L. J. Opalski. Algorithms and Software Tools for IC Yield Optimization Based on Fundamental Fabrication Parameters. *IEEE Trans. on Computer-Aided Design*, 5(1):79–89, Jan. 1986.

[112] Synopsis. *Circuit Explorer Analysis, Optimization and Trade-offs.* 2007. `http://www.synopsys.com/products/mixedsignal/hspice/circuit_explorer_ds.pdf`.

[113] A. Torralba, J. Chávez, and L. G. Franquelo. FASY: A Fuzzy-Logic Based Tool for Analog Synthesis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(7):705–715, July 1996.

[114] C. Toumazou and C. A. Makris. Analog IC Design Automation: Part I—Automated Circuit Generation: New Concepts and Methods. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(2):218–238, Feb. 1995.

[115] G. Van der Plas, G. Debyser, F. Leyn, K. Lampaert, J. Vandenbussche, G. G. E. Gielen, W. Sansen, P. Veselinovic, and D. Leenaerts. AMGIE—A Synthesis Environment for CMOS Analog Integrated Circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 20(9):1037–1058, Sept. 2001.

[116] P. Vancorenland, C. De Ranter, M. Steyaert, and G. Gielen. Optimal RF design using Smart Evolutionary Algorithms. In *IEEE/ACM Design Automation Conf.*, pages 7–10, Los Angeles, June 2000.

[117] P. Vancorenland, G. Van der Plas, M. Steyaert, G. Gielen, and W. Sansen. A Layout-aware Synthesis Methodology for RF Circuits. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 358–362, San Jose, Nov. 2001.

[118] J. P. Vanderhaegen and R. W. Brodersen. Automated Design of Operational Transconductance Amplifiers using Reversed Geometric Programming. In *IEEE/ACM Design Automation Conf.*, pages 133–138, San Diego, June 2004.

[119] I. Vassiliou, H. Chang, A. Demir, E. Charbon, P. Miliozzi, and A. Sangiovanni-Vincentelli. A Video Driver System Designed Using a Top-Down, Constraint-Driven Methodology. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 463–468, San Jose, Nov. 1996.

[120] I. Vassiliou, K. Vavelidis, T. Georgantas, S. Plevridis, N. Haralabidis, G. Kamoulakos, C. Kapnistis, S. Kavadias, Y. Kokolakis, P. Merakos, J. C. Rudell, A. Yamanaka, S. Bouras, and I. Bouras. A Single-Chip Digitally Calibrated 5.15–5.825-GHz 0.18-$\mu$m CMOS Transceiver for 802.11a Wireless LAN. *IEEE Journal of Solid-State Circuits*, 38(12):2221–2231, Dec. 2003.

[121] B. Vaz, N. Paulino, J. Goes, R. Costa, R. Tavares, and A. Steiger-Garção. Design of Low-Voltage CMOS Pipelined ADC's using 1 pico-Joule of Energy *per* Conversion. In *IEEE Int. Symp. on Circuits and Systems*, volume 1, pages 921–924, Scottsdale, May 2002.

[122] M. Vogels and G. Gielen. Architectural Selection of A/D Converters. In *IEEE/ACM Design Automation Conf.*, pages 974–977, Anaheim, June 2003.

[123] G. Wolfe and R. Vemuri. Extraction and Use of Neural Network Models in Automated Synthesis of Operational Amplifiers. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):198–212, Feb. 2003.

[124] Xin Li, P. Gopalakrishnan, Yang Xu, and L. T. Pileggi. Robust Analog/RF Circuit Design with Projection-Based Posynomial Modeling. In *IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 855–862, San Jose, Nov. 2004.

[125] Y. Xu, K.-L. Hsiung, X. Li, I. Nausieda, S. Boyd, and L. Pileggi. OPERA: OPtimization with Ellipsoidal uncertainty for Robust Analog IC design. In *IEEE/ACM Design Automation Conf.*, pages 632–637, Anaheim, June 2005.

[126] L. Zhang, R. Raut, Y. Jiang, and U. Kleine. Placement Algorithm in Analog-Layout Designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):1889–1903, Oct. 2006.

[127] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Trans. on Evolutionary Computation*, 3(4):257–271, Nov. 1999.