ADAPT: Design assistance for iterative analog synthesis

Chieh Lin, Tino Heijmen, Jan ter Maten, Marq Kole
Philips Research Laboratories, Eindhoven, The Netherlands; Email: chieh.lin@philips.com

Abstract

This paper presents an analog circuit design assistance tool called Adapt. The tool can perform automatic circuit sizing over a wide range of heterogeneous design parameters. We present Adapt's architecture and the design flow, discuss important user-interface aspects of the tool, disclose details on the optimization algorithms incorporated in the tool, and demonstrate the practical strength of the tool. The industrial-strength capabilities of the tool are showcased using an RF low-noise amplifier that is used in Bluetooth applications.

1 Introduction

Although the current trend in integrated circuit design is to implement as much as possible of the functionality of an integrated system in the digital domain, analog circuits are still indispensable to interface to the real world. A clear example is the development in RF applications for mobile and wireless communications. Apart from this, the design of any circuit at the transistor level is basically analog circuit design and that includes the design of digital standard cells used in digital circuitry.

Designing analog and RF building blocks is a time-consuming and complicated task. Starting from a number of specifications, a designer has to use his experience and creativity to select or develop a topology that might be able to meet those specifications. From the chosen circuit topology he has to determine how and under which conditions the specifications can be met. This is normally accomplished by selecting the right values for all components available in the circuit. In the design of integrated circuits the designer has relatively much freedom in the selection of possible values through the related geometrical dimensions of the components, for instance the width and length of MOS transistors and poly resistors, the emitter area of bipolars, etc. Therefore this is also called the sizing of the design. Next to component values also the values of bias currents and voltages (including the supplies) should be considered design parameters.

Manual sizing of the design is a tedious task, but gives a lot of insight into circuit behavior, which might be useful at a later stage of the design process. As the human designer knows which specifications are the most important or the most difficult to meet, he can keep possible design solutions in mind for every specification. The implications of this design process are clear. Firstly, analog design requires a lot of experience of the designer. Secondly, automating this task is both knowledge and computation intensive. Still, the main reason one would like to automate this task is time reduction of analog building block development. Also, with the scarcity of experienced analog designers one would like such a designer to focus on the creative part of analog circuit design and leave the tedious and less creative part of it to a computer.

In the past 15 or so years quite a number of developments into the area of analog synthesis have appeared. Where initial efforts focused on the automation of the complete analog design process, the more recent results offer help at particular points in the process: circuit sizing, design centering, design space exploration. Other important aspects of analog synthesis are automated topology selection and layout generation. Although a lot of progress has been made in the past decades, tools in the latter areas are not yet mature enough, probably due to more difficult challenges, compared to circuit optimization tools. In this paper we employ analog synthesis to cover only circuit optimization.

Within our group a tool called Adapt has been developed which focuses on circuit sizing. In Section 2 we present Adapt in the context of previous analog synthesis developments. We will discuss its interaction with the user, which is a vital point of each tool; how does the user enter his design problem, how does he interact with it, and how is it embedded in the complete design flow. In Section 3 we reveal which optimization algorithms Adapt uses for its operation. A new algorithm, developed for Adapt, is discussed in detail. Section 4 demonstrates the true power of Adapt; a demanding contemporary circuit has been optimized and, consequently, has led to a considerable improvement compared to manual design. Finally, in Section 5 we wrap up with a few conclusions and directions for future research.

2 Adapt: Analog Synthesis

Whereas digital design is highly automated and relies heavily on synthesis tools, the bulk of analog design is done by hand and requires a significant amount of design effort even though analog parts commonly occupy only a relatively small part of the silicon area in the final design. The main reason for this design effort is that analog design is knowledge-intensive due to the strongly non-linear nature of the performance measures and the high sensitivity of these measures to variations in the design parameters, resulting in design problems with complex trade-offs. In order to decrease the design time for analog design and to be able to handle in-

creasingly complex circuits and design processes, significant effort has been spent during the past decades on the automation of analog design [4].

In the past, a number of approaches have been proposed to deal with the problem of automatically sizing the circuit, sometimes in combination with automated topology selection. These approaches can be roughly categorized into knowledge-based methods, using design knowledge and heuristics, and optimization-based methods, applying numerical programming techniques.

The first generation of knowledge-based tools for analog circuit synthesis were presented in the second half of the 1980s; programs such as BLADES [3], IDAC [1], and OASYS [5]. However, all these programs suffered from the problem that the heuristic knowledge of an analog designer is difficult to acquire explicitly. Not only was it a very time-consuming process to encode design knowledge for a given set of specifications, but this design knowledge also had a limited lifetime. The high rate of progress in process technologies made knowledge acquired yesteryear less an obviously good choice for current technologies. Therefore, the application of knowledge-based analog synthesis approaches has been limited.

Starting from the late 1980s a second generation of methods emerged that apply optimization techniques to determine the values of the design parameters in order to optimize the circuit performance for a given set of specification constraints. These methods are more flexible than the knowledge-based approaches and can be extended more easily to new circuit types. Two subcategories can be distinguished: the equation-based and the simulation-based approaches. The equation-based methods use analytic design equations to describe the circuit performance. The advantages of the equation-based approach are the short evaluation time and the flexibility. The main drawback is that analytical models have to be used to derive the design equations and, despite recent progress in symbolic circuit analysis, not all design characteristics can be easily captured in analytic equations with sufficient accuracy. These problems can be avoided by applying a simulation-based approach where a circuit analysis tool is executed in the inner loop of the optimization to determine the circuit performance. Although this used to be computationally too expensive, with the exponential increase of computation power this approach has become increasingly favorable. Especially for small and medium-sized circuits the relatively high evaluation cost of numerical simulation is not an obstacle. Tools such as DELIGHT.SPICE, FRIDGE, ASTRX/OBLX, MAELSTROM, and ANACONDA (see [4] for an overview) are examples of simulation-based tools.

Thus, although automated circuit sizing is not by far as commonly used as, for example circuit simulation, it has gained increasing interest in the analog design process during the last decades. However, a key difficulty is that the analog design problem, with all the involved design knowledge and heuristics, has to be formulated as an optimization problem such that the final optimized results are acceptable to the designer. This often presents a high threshold for using a circuit-sizing tool.

2.1 Adapt Flow

In Fig. 1 the Adapt design cycle is shown as a continuous loop. It is assumed that the designer has selected an appropriate circuit topology, i.e. it might be able to fulfill all specifications once properly sized ("unoptimized netlist"). At the top left side of the figure, the designer enters the design cycle by defining the circuit performance targets ("circuit characteristic", "operator", "target"). Also, each of these circuit specifications have to be put in one or more priority groups, which is explained shortly.

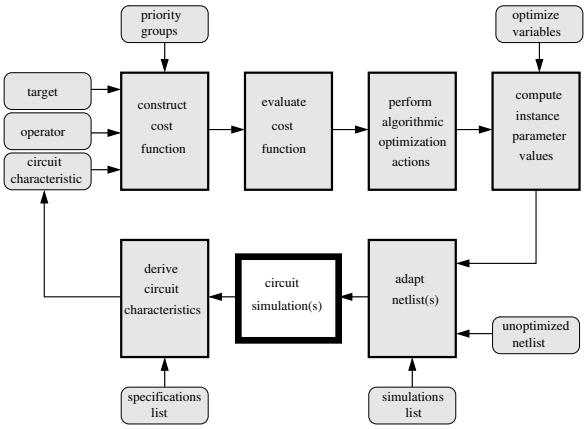

Fig. 1: The Adapt design cycle.

Once these inputs have been handled, Adapt starts to construct the cost function for this particular optimization problem. Based on sampled values around the given initial solution, the optimization routine will determine the most likely direction of a better solution and provide a new set of values for the optimize variables. Optimize variables are the basic parameters the optimization routine operates on. It is better to use optimize variables rather than the design parameters directly as this will allow a designer to specify a number of design parameters that are correlated, for instance for transistor matching purposes. This approach clearly reduces the number of independent optimize parameters which has a positive effect on overall run-time. From the optimize variables, the design parameters (instance parameter values) are calculated. These can now be inserted in the original netlist at the appropriate positions. Next, the designer needs to enter a so-called simulations list: a list of simulations to be carried out for the circuit to be able to obtain the simulation data necessary to calculate the design characteristics. This might be a number of transient simulations for a digital cell, but may also be a number of complicated RF simulations for an RF mixer circuit or a VCO. Using this simulations list, a set of netlists is produced

and the simulation job is fed to the simulator. Using the user-defined specifications list, Adapt will derive from the simulation output all required information to compute the circuit characteristics and quantify them in terms of the given specifications. The comparison of the design characteristics with the design specifications is done through a new evaluation of the cost function. The quality of the design characteristics is quantified by the cost function value and this, in turn, guides the optimization algorithm in choosing the next adaptation of the optimize variables. Now we have completed a single iteration of the Adapt design cycle. Typically the optimization algorithm iterates this process until a predefined number of evaluations has been performed, (non-)convergence has been detected or the user has stopped the program.

Adapt can be highly interactive. It gives feedback in a graphical form and easily allows for exploration of any circuit topology, enabling the designer to build up design knowledge when working with the tool. Adapt uses an external numerical circuit simulator and a derivative-free nonlinear constrained optimization algorithm.

Even though the underlying methods in Adapt are optimization methods, the program hides the mathematical complexity implied by these methods from the user. This is achieved by automatically formulating the optimization problem in a similar way as is done in a manual design process. It provides a user interface that uses circuit characteristics and design specifications to communicate with the user. This interface is highly interactive and has been designed to be as close as possible to the design environment the designer is comfortable with. It leaves the maximum amount of freedom to control the design process as desired.

One aspect of optimization that often proves a great barrier to entry of potentially interested designers is the formulation of a mathematical optimization problem. This involves setting up a cost function, which is specific for each problem and which guides the optimization in the desired direction. Adapt provides automated generation of the cost function based on a simple and intuitive set of operators, familiar to analog designers. The current approach, evolved from many years of user-feedback, is intuitive and straightforward, hiding as many optimization-specific details as possible.

To analog designers some specifications are more important than others. For instance, a functional requirement such as phase margin has higher priority than minimizing the power consumption. Only once the specification of higher importance has been satisfied should the sizing process try to meet the specification of the lower priority item. This process has been implemented in Adapt through so-called priority groups. The optimization algorithm will try to meet the specifications in the higher priority groups first, and optimizes the lower priority groups while constraining the achieved higher priority specifications not to fall below their accepted values for the remainder of the optimization process. In practice the designer will put the functional requirements at higher priority groups and global optimization for power and area in the lower priority groups.

2.2 Adapt User Interface

Adapt intends to help an analog designer tune a circuit with a given topology to fit its specifications. An important perceived benefit is that it allows the designer to use his own experience to guide the tool in the right direction and arrive at a design that meets the specifications. The main contribution required from the designer is simulation and design knowledge. The program is a design assistant that gives the user full control over the design, the specifications, and the way in which the specifications should be met.

To measure the performance of the design, a so-called test-bench is required. This test-bench is made up from three parts: a test-rig, the specifications list, and the simulations list. The latter two were discussed with the Adapt design cycle; the first requires some explanation. A test-rig consists of extra circuitry (such as voltage and current supplies, loads at outputs, bias sources, etc.) that is added to the selected circuit topology in order to allow simulations specified in the simulations list to be performed.

An interesting application of Adapt is the migration of an existing design to a new process technology. For this approach the test-bench need not be changed. Indeed, it is the centerpiece that allows one to test the specifications for the new design in the same way that they were derived for the old design. It is generally a good idea to construct a test-bench in a process-independent manner, keeping reuse in mind.

An important consideration for analog synthesis is that the design process mimics as much as possible the design style and environment the user is comfortable with. That means integration into the common design environment (such as Cadence Design Framework II), support of a simulator qualified for the job (an industrial-strength simulator such as Avant! Star-Hspice), and access to all necessary output facilities the user would use during the similar manual process.

At the moment both the Philips in-house analog circuit simulator Pstar can be used as well as Cadence's Spectre simulator. Aimed at the design of small or medium-sized building blocks such as opamps, comparators, logic gates, VCOs, mixers, Adapt has already been applied successfully at several design centers throughout Philips Semiconductors.

3 Optimization Algorithms

Adapt transforms a design problem into a series of optimization problems. Candidate algorithms for solving

these have to satisfy a number of criteria:

- The algorithm should be capable of handling nonlinear constrained optimization problems.
- The number of required function evaluations should be as low as possible, because a full circuit simulation is performed in each optimization step.
- The algorithm should be truly derivative-free. Derivatives of the performance functions with respect to changes in the design variables are not provided by the simulator. Finite-difference schemes cannot be used, because the function values are contaminated by numerical noise.

Adapt applies two different optimization algorithms that satisfy these conditions. The first combines the Nelder–Mead (NM) method for unconstrained optimization with a quadratic penalty function to include constraints. The second uses an augmented Lagrangian as a merit function, which is minimized by a grid-based trust-region approach. The latter algorithm has been developed specifically for Adapt.

The search for the optimal values of the optimization variables (OVs) can be formulated as a nonlinear constrained optimization problem in n variables with m constraints,

$$\begin{array}{lll} \text{minimize} & f(\mathbf{x}), & \mathbf{x} = (x_1, x_2, \ldots, x_n), \\ \text{subject to} & c_i(\mathbf{x}) \leq 0, & i = 1, \ldots, m, \\ & a_i \leq x_i \leq b_i, & i = 1, \ldots, n, \end{array} \tag{1}$$

where x_i denotes the ith OV, with lower and upper bounds a_i and b_i, respectively. The values of the objective function $f(\mathbf{x})$ and the constraint functions $c_i(\mathbf{x})$ are obtained from circuit simulation. Equality constraints are not included in Eq. (1) because Adapt needs and supports only inequality constraints. However, the algorithms could be easily adjusted to accommodate to problems including equality constraints.

The performance and stability of the optimization algorithms are affected by the *scaling* of the OVs and the values of the merit functions. Several transformation types are of interest in a circuit sizing tool. Linear transformation is most suitable for OVs that vary only over a small range. On the other hand, OVs with large ranges are best scaled by a logarithmic transformation. However, if the variable or function is not strictly positive or negative, a transformation has to be defined that behaves logarithmically in the infinite limit, but that can also be applied to variables of indefinite sign. This kind of transformation is called quasi-logarithmic. All three scaling types are used in Adapt.

The NM algorithm has been widely used in practical applications because of its simplicity and robustness. However, it suffers from several drawbacks, such as moderate performance, lack of a solid theoretical basis, and danger of degeneracy of the simplex. Because the total process time is mainly determined by the number of evaluations, the relatively poor performance of the NM algorithm can be a problem in the optimization of larger designs. Therefore, an alternative algorithm, named Gridmom, was developed for application in Adapt. This algorithm is treated below. For a more detailed discussion we refer to a forthcoming paper [6].

3.1 Method of Multipliers

The Gridmom algorithm applies a sequential minimization of an *augmented Lagrangian* penalty function. By introducing a slack variable $s_i \geq 0$, each inequality constraint in Eq. (1) can be rewritten as an equality: $c_i(\mathbf{x}) + s_i = 0$. The augmented Lagrangian penalty function can then be written as,

$$\Phi_{\mathrm{ALAG,s}}(\mathbf{x}; \boldsymbol{\lambda}; \boldsymbol{\mu}; \mathbf{s}) = f(\mathbf{x}) + \sum_{i=1}^{m} \left\{ \lambda_i [c_i(\mathbf{x}) + s_i] + \mu_i [c_i(\mathbf{x}) + s_i]^2 \right\}, \tag{2}$$

where the parameters λ_i and μ_i are Lagrange multipliers and penalty factors, respectively. Minimization over the slack variables s_i yields a simplified penalty function,

$$\Phi_{\mathrm{ALAG}}(\mathbf{x}; \boldsymbol{\lambda}; \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{m} \left\{ \mu_i \max\left[c_i(\mathbf{x}) - \frac{\lambda_i}{2\mu_i}, 0 \right]^2 - \frac{\lambda_i^2}{4\mu_i} \right\}. \tag{3}$$

In contrast with the quadratic penalty function, an exact solution of the augmented Lagrangian can be found for finite values of μ_i, provided that these values are sufficiently large and that the values of λ_i are simultaneously optimized. If these conditions are met, optimizing the penalty function of Eq. (3) results in a combined set of variable values x_i^* and multipliers λ_i^* that correspond to a solution of Eq. (1), independent of the chosen values of μ_i.

The algorithm uses the *method of multipliers* (MOM) to solve the problem of Eq. (1), with the augmented Lagrangian of Eq. (3) as a merit function [6]. After a counter k is initialized ($k = 1$), the variables x_i and the penalty factors μ_i are set to their initial values $x_i^{(0)}$ and $\mu_i^{(0)}$, respectively, and the multipliers λ_i are set to $\lambda_i^{(0)} = 0$. The values of λ_i and μ_i are fixed and the merit function,

$$\Phi_{\mathrm{ALAG}}^{(k)}(\mathbf{x}) = \Phi_{\mathrm{ALAG}}(\mathbf{x}; \boldsymbol{\lambda} = \boldsymbol{\lambda}^{(k-1)}; \boldsymbol{\mu} = \boldsymbol{\mu}^{(k-1)}), \tag{4}$$

is minimized, resulting in the argument $\mathbf{x}^{(k)}$ for which the function is minimal,

$$\mathbf{x}^{(k)} = \arg\min_{x} \Phi_{\mathrm{ALAG}}^{(k)}(\mathbf{x}). \tag{5}$$

The values of the multipliers are updated using $\mathbf{x}^{(k)}$,

$$\lambda_i^{(k)} = \lambda_i^{(k-1)} - 2\max[\,\mu_i^{(k-1)} c_i(\mathbf{x}^{(k)}), \lambda_i^{(k-1)}\,]. \tag{6}$$

The updated multiplier values define a new merit function. The value of k is incremented and $\Phi^{(k)}_{\mathrm{ALAG}}(\mathbf{x})$ is again minimized, giving a new solution $\mathbf{x}^{(k)}$. Then, *either* the multiplier λ_i is updated, according to Eq. (6), *or* the corresponding penalty factor μ_i is increased. Which of the two parameters is adjusted depends on the violation of the corresponding constraint $c_i(\mathbf{x})$. The sequence of minimizing the merit function of Eq. (3) and updating the parameters λ_i and μ_i is repeated until the termination criteria have been satisfied.

A combined set of variable values $\mathbf{x}^*$ and multipliers $\boldsymbol{\lambda}^*$ is regarded as a solution of Eq. (1), if the first-order Karush–Kuhn–Tucker (KKT) conditions are fulfilled. The algorithm is also ended if the maximum allowed number of subproblems has been exceeded without finding a solution. It is then left to the user to accept the intermediate results or to modify the input specifications and restart the optimization.

3.2 Trust-Region Minimization

The original nonlinear constrained problem is converted into a sequence of bound-constrained subproblems, as discussed above. Each subproblem involves the minimization of a merit function $\Phi^{(k)}_{\mathrm{ALAG}}(\mathbf{x})$. To this purpose, an algorithm is applied that is similar to the one reported in [2]. This algorithm uses a grid and applies a trust-region approach that approximates $\Phi^{(k)}_{\mathrm{ALAG}}(\mathbf{x})$ by a *quadratic model function*,

$$q(\mathbf{x}) = a + \mathbf{g}^T(\mathbf{x}-\mathbf{x}^{\mathrm{ref}}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^{\mathrm{ref}})^T\mathbf{G}\,(\mathbf{x}-\mathbf{x}^{\mathrm{ref}}), \tag{7}$$

which is minimized within a *trust-region* $\mathcal{B}$ with radius Δ centered at the reference point $\mathbf{x}^{\mathrm{ref}}$,

$$\mathcal{B} = \{\, \mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}-\mathbf{x}^{\mathrm{ref}}\|_\infty \le \Delta \,\}. \tag{8}$$

In this region, $q(\mathbf{x})$ is assumed to be a good estimate of the true merit function. Because the infinite norm $\|\mathbf{x}\|_\infty \equiv \max_i |x_i|$ is used, the trust-region has a square or (hyper-)cubic shape. Evaluation of the merit function is restricted to points on a grid. Initially the grid is relatively coarse, but it is refined during the optimization process. The application of successively refined grids prevents the clustering of evaluation points in an early stage.

The algorithm for minimizing $\Phi^{(k)}_{\mathrm{ALAG}}(\mathbf{x})$ distinguishes three phases: a *starting phase*, a *descent phase*, and a *refinement-check* phase. After the starting phase, the descent and refinement-check phases are alternately executed until the termination criteria have been fulfilled. The details of the starting and descent phases primarily determine the progress of the algorithm. On the other hand, the properties of the refinement-check phase are mainly responsible for its robustness and convergence behavior.

During the *starting phase*, the merit function is evaluated at as many points as are necessary to enable the construction of an approximating function. The Gridmom algorithm includes two different starting phase methods: one applying the Hooke–Jeeves (HJ) algorithm, the other using the Uniform Design (UD) approach. The second method is based on number theory and generates a uniform sample point distribution in the feasible domain [6].

The trust-region approach is applied in the *descent phase*, where the model function $q(\mathbf{x})$ is constructed by least-squares fitting to a set of points at which the merit function has previously been evaluated. The selection criterion for the points in this set is the distance from the reference point. The descent phase is entered as soon as possible, because progress is assumed to be faster than during the starting phase. Initially, the descent phase applies a linear approximating function. When more evaluations have been performed and the number of available fitting points increases, the linear model is replaced by a quadratic model with a diagonal Hessian $\mathbf{G}$. When sufficient evaluation points are available, the algorithm switches to a quadratic function with a full Hessian.

An accurate estimate of the gradient $\mathbf{g}$ is more crucial than an accurate approximation of the Hessian $\mathbf{G}$ in Eq. (7). Therefore, the accuracy of the gradient is improved locally by performing a second least-squares fit, with a fixed Hessian obtained from the first fit, using a small number of evaluation points close to $\mathbf{x}^{\mathrm{ref}}$.

The approximating function is minimized within the trust-region and the merit function is evaluated at the feasible grid point nearest to the minimum of the approximating function. The trust-region radius and the reference point are updated and a new approximating function is constructed. This process is repeated until either a grid point is proposed at which the merit function has already been evaluated or until a maximum number of successive iterations have not resulted in an improved estimate of the optimum.

The update of the trust-region radius Δ is based on the change in the value of the true merit function compared to the reduction predicted by the approximating function. The criterion for updating the reference point $\mathbf{x}^{\mathrm{ref}}$ is determined by a similar ratio.

The *refinement-check phase* determines whether the grid should be refined. To this purpose a linear approximating function is fitted to a set of nearby evaluation points and minimized. If necessary for the construction of a non-singular linear function, the merit function is evaluated at a number of additional points. The algorithm returns to the descent phase if the grid point closest to the minimum of the linear model is a better estimate of the optimum than the current approximation. Otherwise, if new evaluation points have been attained during the refinement-check phase, a quadratic model function is constructed and minimized within the trust-region $\mathcal{B}$ of Eq. (8), just as in the descent phase. If these efforts do not result in a better estimate of the optimum, the grid spacing is reduced.

The current estimate of the optimum is accepted as the solution of the subproblem when a *stationary point* of the merit function has been obtained. Criteria with respect to both the variable values and the function values of the augmented Lagrangian are applied to test for a stationary point. The algorithm is also terminated after a maximum number of function evaluations or when the grid has been refined a maximum number of times.

Although each minimization of the current augmented Lagrangian is an optimization problem on its own, it is possible to *reuse* the values of the objective and constraint functions obtained when solving previous subproblems. The values of $f(\mathbf{x})$ and $c_i(\mathbf{x})$ computed for previous evaluation points are stored (an evaluation point is defined as a set of OV values). This information is used to compute the value of the current augmented Lagrangian for the stored evaluation points, given the current values of the Lagrange multipliers λ_i and the penalty factors μ_i. Thus, a starting set of data for the construction of approximating functions is available after the first subproblem has been solved. This reuse of simulation data can significantly increase the performance of the MOM approach.

4 A Design Example

In this section we demonstrate the strength of Adapt by discussing some of the results we have obtained recently on contemporary circuits. The demonstration vehicle is an RF low-noise amplifier that is used in Bluetooth applications.

Due to intellectual property constraints, we are unable to disclose details on the circuit topology.

4.1 RF Low-Noise Amplifier

Every analog circuit designer knows that typically many specification trade-offs are possible, some of which are exceptionally attractive. This case shows that Adapt can successfully find one of these special cases for an RF low-noise amplifier circuit.

While we increase 40% of the LNA power consumption (including a mirror suppression filter), we can gain more than 3dB in noise figure value, obtain 2dBv increase in gain, and reduce third order harmonic distortion by more than 3dBv. This is a significant improvement of the manual design.

This design has 14 optimize variables, 34 design parameters, and 4 specifications for gain, power, linearity and noise. Although the number of optimize variables is quite modest, this particular design required the use of advanced RF analysis methods that are much more computationally expensive than conventional analyses. Therefore, it might be considered a challenge to formulate the optimization problem in such a way as to find a good solution in a reasonable amount of time. From the experiments, we observed that Adapt (using Gridmom) is very robust and the overall run-time does not increase dramatically as the problem size increases (number of specs, number of optimize variables, number of simulations). The optimization run-time was somewhat more than one hour for 111 simulation cycles (see Fig. 1) on a standard HP9000/800 UNIX server.

5 Conclusions and Directions

We have shown that Adapt is a true analog design assistance that can help a designer find better circuit solutions in less time. Adapt is very well integrated in the overall design flow using the Cadence DF II environment. It exhibits a friendly graphical user interface that shields the designer from non-relevant algorithmic aspects. Consequently, setting up the tool for a specific design problem is quite straightforward. The new Gridmom algorithm in Adapt has been discussed. Moreover, the industrial-strength power of Adapt has been demonstrated by the improved result of an RF low-noise amplifier for Bluetooth applications.

There are still points for improvements that we are considering. Some of those are: efficient determination of solution robustness and better handling of heterogeneous test-benches. Last but not least, taking into account layout effects during optimization is of major importance, especially for RF circuits.

References

[1] Degrauwe, M.G.R., Goffart, B.L.A.G., Meixenberger, C., Pierre, M.L.A., Litsios, J.B., Rijmenants, J., Nys, O.J.A.P., Dijkstra, E., Joss, B., Meyvaert, M.K.C.M., Schwarz, T.R., Pardoen, M.D., "Toward an analog system design environment", *IEEE J. SSC*, Vol. 24 (1989), pp. 659–671.

[2] Elster, C., and Neumaier, A., "A Grid Algorithm for Bound Constrained Optimization of Noisy Functions", *IMA J. Numer. Anal.*, Vol. 15 (1995), pp. 585–608.

[3] El-Turky, F., and Perry, E.E., "BLADES: an artificial intelligence approach to analog circuit design", *IEEE Trans. CAD*, Vol. 8 (1989), pp. 680–692.

[4] Gielen, G.G.E., and Rutenbar, R.A., "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits", *Proc. IEEE*, vol. 88 (2000), pp. 1825–1852.

[5] Harjani, R., "OASYS: a framework for analog circuit synthesis", *IEEE Trans. CAD*, Vol. 8 (1989), pp. 1247–1266.

[6] Heijmen, T.G.A., Kevenaar, T.A.M., Pranger, H.-J., "Adapt: an Optimization-Based Analog Design Assistance Tool", submitted for publication.

[7] Kole, M., Heijmen, T., Kevenaar, T., Pranger, H.-J., Sevat, M., "Adapt, an interactive tool for analog synthesis", *Proc. SAME Conf.*, Sophia Antipolis, France, 2001.