Chapter 11

# Design of Analog Integrated Circuits Using Simulated Annealing/Quenching with Crossovers and Particle Swarm Optimization

Tiago Oliveira Weber and Wilhelmus A. M. Van Noije

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50384

## 1. Introduction

Electronics have received great advance in manufacturing technology and in design automation in the last decades. Systems-on-chip (SoC) integrating complex mixed-signal devices with multi-million transistor circuits have been developed. This fast increase in complexity has been reflected on the need for more engineers and tools to increase productivity.

In the digital domain, the tools are comprehended inside a well-defined design flow that allows the designer to segment the problem in various abstraction levels and then solve it in a straightforward approach. There are industry standards for digital tools in this domain and new tools can be proposed to solve specific problems and be allocated properly in the flow. In analog domain, despite of the efforts in the development of design tools by the scientific community, there is still no well-defined flow due to the great complexity and the interrelation between the several phases of design. Therefore, there is a gap in the automation level of flows between digital and analog domain. A greater automation of the analog flow would allow smaller time-to-market for analog design and also for mixed-signal designs, as analog sections are usually the bottleneck in time required to design.

Analog circuits have an important role in most of modern ICs. They are used in the interface between real world signals (which are analog) and the digital world, covering applications from analog-to-digital and digital-to-analog converters to filters and radio frequency circuits. Therefore, productivity in the analog domain reflects on the IC industry as a whole.

The difficulties of automating analog designs result from the lack of clear separation of stages in the analog flow. Altought it is possible to describe the flow using some abstract steps, in practice the flow is strongly connected through feedback and low-level design tasks need to be considered even when the design is at a high-level stage. Therefore, a complete analog design is usually not a straightforward process, requiring comprehension of the whole process in order to perform a single task.

Other issue that makes the automation in the analog domain more complex than in the digital is the great number of degrees of freedom in the design variables, as well as the lack of clear interrelation among them. Also, analog variables are within an infinite range of values and the measurements performed need to take into account second and third-order effects. This means that the trade-offs that exist are not always evident and therefore very experienced designers are needed to have the correct insights during the sizing of analog blocks.

Furthermore, once the design is accomplished for a selected technology, new effort and time is required to design it for a new technology.

As changing technology usually means that the project has to start from scratch, an expert analog designer (or a team of designers) is required once again as a change of technology usually results in redesign from scratch. Despite all difficulties, analog design is typically accomplished by engineers with no more than a mathematical software, an electrical simulator, intuition and energy to perform several trials and errors, as no hand calculation delivers precise results in one attempt.

The use of CAD tools is a way to aid the designer to bridge the productivity gap between the analog and digital domains. Synthesis is optimization procedure which does not rely on an initial good solution. It has the objective of achieving the specifications without the requirement of an initial solution close to the global minimum provided by the designer.

Initial studies in circuit level synthesis were done in the 1980's and were mostly knowledge-based approaches [5], [10]. These approaches tried to capture the design knowledge required to perform some circuit implementation and implement it in algorithms. Optimization-based approaches started to be made in the late 1980's and are still an active area of research [6], [29], [27]. These approaches provide tools with greater flexibility and smaller overhead in comparison with knowledge-based approaches.

Optimization-based approaches are further classified by the way they evaluate a new solution. Simulation-based evaluation rather than equation-based evaluation was progressively being adopted in order to reduce the preparatory effort and to provide higher accuracy. This type of evaluation is benefited from the advances in computer power which allow faster simulations and therefore can return precise synthesis results within a tolerable amount of time.

This chapter will introduce the basics involved in using Simulated Annealing/Simulated Quenching as the optimization core to synthesize analog integrated circuits, as well as the use of this algorithm together with population-based algorithms in order to improve the effectiveness of the results. The use of multi-objective information to combine Simulated Quenching with Genetic Algorithms as well as the use of Simulated Quenching followed by Particle Swarm Optimization is performed to show the advantages of these techniques. Population-based algorithms are better in dealing with multi-objective problems as they can use pareto-dominance to compare solutions. Combining SA/SQ with population-based algorithms allow the use of SA/SQ qualities with the possibility of overcoming issues related to the use of aggregate objective functions (used to convert multi-objectives into one) . Multi-objective information can guide the algorithm when it is locked in a local minimum and also used combined with SA to explore the pareto-front. In this work, all synthesis were performed using the software *MATLAB* for processing and *HSPICE* for electrical simulations.

The chapter is divided as follows. In section 2, the definition of the analog design problem is presented. Section 3 presents a brief history of the analog synthesis approaches which used

Simulated Annealing. In section 4, the algorithms for SA/SQ and the combinations with GA and PSO are shown. Section 5 presents synthesis of two amplifiers and of a voltage reference using the developed algorithms. Finally, section 6 presents the conclusions.

## 2. Analog design overview

Analog design is a complex task which involves knowledge at different levels of abstraction and the use of one or more appropriate tools for each level. At the system level, the designer must select the architecture that best fits the analog system that will be developed. This level describes the lower levels simply as functional blocks, which have not yet direct connection with the electrical devices that will implement them. The circuit level is responsible for converting these functional blocks in electronic devices. Each high level block will have a circuit level. The topology and size of the devices in this level will define the specifications which can be used on the system level. The layout level is responsible for converting the dimension values from the circuit level to the mask designs which will later be used in the fabrication process. Figure 1 illustrates the abstraction levels involved in analog design.

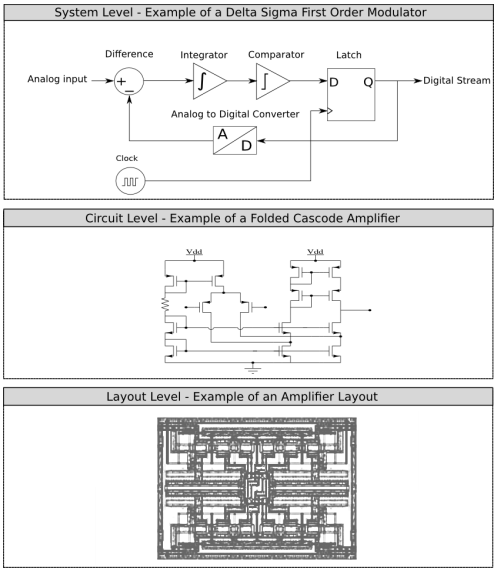


**Figure 1.** Different Levels (abstractions) of Analog Circuit Design.

The focus of this chapter will be restrained to analog design at circuit level, which is the problem of finding the best device dimensions (transistors, resistors, capacitors,...) and electrical variables (voltages and currents) in order to achieve the requirements for each block. In this perspective, the parameters used in the system level for each block are the specifications in the circuit level. The performance metrics of the circuit must be attended through the manipulation of the dimension variables of the devices.

This is a very challenging problem as it has multi-objectives, has several local minimums and the time required to evaluate each solution is often not negligible if a simulation-based approach is used. The objectives of an analog synthesis and the performance constraints are

usually non-linear functions of circuit parameters. In [2] is presented a list of desirable features in an analog synthesis tool. This list includes small preparatory overhead, small synthesis time, starting point independence, accuracy, variation-tolerance, generality and fabrication process independence.

The preparatory overhead is the time spent before the optimization starts in which the designer converts the problem to the format understandable by the optimization tool. In equation-based evaluations, this overhead is usually high (weeks to months) and demands an experienced engineer to be performed properly. In simulation-based evaluations the overhead is reduced as the simulator will be the responsible for extracting the measurements from the proposed solutions. The designer usually only have to provide the circuits in a spice-like format.

Once all setup is ready, synthesis time is much faster in equation-based approaches. However, with the rapid increase in computer power the time required for simulation-based approaches is also being strongly reduced.

Starting point independence refers to being able to achieve good solutions without the requirement of an initial good solution provided by the designer. This is a fundamental requirement for characterizing a procedure as synthesis and justifies the use of global search techniques instead of using less complex local search techniques without capability of avoiding local minimums.

Accuracy is often a problem only on equation-based evaluation techniques, as the design model needs to capture all important effects on the devices and still provides equations manageable by the optimizer. Simulation-based approaches use transistor models like BSIM3 and BSIM4 which offer a good trade-off between accuracy and evaluation time.

Variation-tolerance is the capability to provide solutions that are not only good in the nominal device operation, but also considering the variations that might occur during fabrication process. Generality refers to being able to synthesize a wide range of circuit types. This is also a characteristic at which simulation-based approaches perform better as the only requirement for this approach is the simulator capability of processing that type of circuit.

Fabrication process independence is the possibility of changing the fabrication technology after the problem is formulated. This is a very important requirement as is not unlikely for a design center to change the fabrication process being used. Then, redesign of its intellectual property (IP) blocks become necessary.

As it can be seen by the desired characteristics, a synthesis tool for analog design needs to address most or all of them in order to become useful to the industry. A tool that requires too much time and effort to be properly used or one that does not returns robust solutions will not fit the designers' expectations.

In figure 2 the circuit level analog synthesizer can be seen as a black-box with its inputs and outputs. The designer provides the topology to be synthesized (e.g., folded cascode, miller amplifier,...), the foundry technology, the specifications (e.g., gain, area, power supply,...), the testbenches which will be used to make the measurements of the circuit and finally the synthesis configurations. The synthesizer interacts several times with the electrical simulator in order to evaluate the solutions that are proposed during the optimization procedure. The outputs are a dimensioned and biased circuit, the specifications achieved by this solution and graphics showing an approximation of its pareto front.

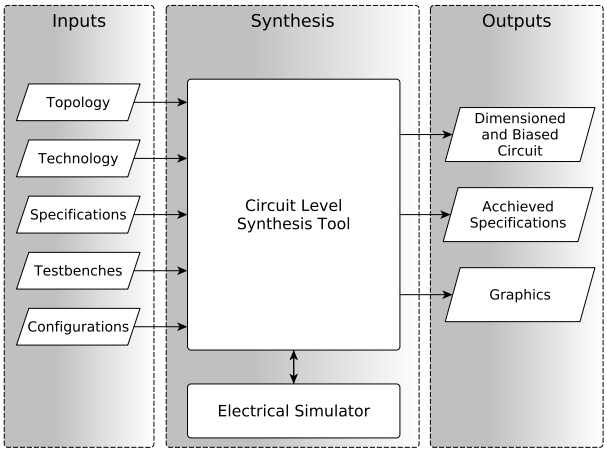


**Figure 2.** Inputs and Outputs of the Circuit Level Synthesis Tool.

## 3. Problem definition

Before we deal with the optimization of analog circuit designs, it is important to spend some effort in correctly defining the problem. An oversimplified definition, considering only the most superficial measurements would either take the solver to a non-practical, non-robust solution or lack enough hints during optimization to direct the solution to the designer specifications.

Analog design at circuit level typically have geometric variables and electrical variables. Geometric variables are the dimensions of the devices present inside a given block, such as the width and length of the transistor. They are important to determine the device characteristics such as the value of a resistor, the value of a capacitor, I/V curve of a transistor and others.

The influence of these geometric variables on the circuit measurements is usually non-linear. The effect of the geometric variables in a transistor current, which will affect the measurements, will depend not only in its dimensions but also on the voltages in its terminals. The equations of a MOS transistor can be seen in equation (1) for long-channel approximation (Schichman-Hodges model):

$$I_D = \begin{cases} 0, & \text{if in cut-off region} \\ \mu_n C_{OX} \dfrac{W}{L}[(V_{GS}\text{-}V_{TH})V_{DS}\text{-}\dfrac{V_{DS}^2}{2}], & \text{if in linear region} \\ \dfrac{\mu_n C_{OX} W}{2L}(V_{GS}\text{-}V_{TH})(1+\lambda V_{DS}), & \text{if in saturation region} \end{cases} \quad (1)$$

where $I_D$ is the drain current, $W$ is the width of the transistor channel, $L$ is the length, $\mu_n$ is the electron mobility, $C_{OX}$ is the gate oxide capacitance, $V_{GS}$ is the gate to source voltage, $V_{TH}$ is the threshold voltage, $V_{DS}$ is the drain to source voltage and $\lambda$ is the channel length modulation effect parameter. From these variables, $W$ and $L$ are the ones which the designer can manipulate while $C_{OX}$, $\mu_n$ and $V_{TH}$ are technology dependent, and the voltages are

consequence of the whole circuit equations (the designer will have indirect control upon these values by setting the correct currents in all transistors).

Electrical variables used by the synthesizer are voltages or currents that are usually in the interface of the block with its external environment, such as bias currents or voltages.

The choice of the variables is not arbitrary. As they will result in physical devices, there are constraints that need to be addressed in order the design to be feasible. These limitations are called design constraints [22]. They can be of two types: behavior (or functional) and geometric (or side) constraints. Behavior constraints are usually provided by the designer to define limitations that guarantee that the circuit will work properly, such as transistor operation regions or relations between different dimensions across transistors.

The operation region of a transistor is the result of the voltages between its terminals as it can be seen in eq. (2), (3) and (4). To get a transistor in a specific operation region is often a problem constraint as this will affect the operation and robustness of the circuit.

$$\text{cut-off} \qquad V_{GS}<V_{TH} \tag{2}$$

$$\text{triode} \qquad V_{DS}<V_{GS}\text{-}V_{TH} \quad \textit{and} \quad V_{GS}>V_{TH} \tag{3}$$

$$\text{saturation} \qquad V_{DS} \geq V_{GS}\text{-}V_{TH} \quad \textit{and} \quad V_{GS}>V_{TH} \tag{4}$$

Geometric constraints are usually provided by the foundry which will fabricate the chip and refer to limitations which narrow the possible widths and lengths. The most common geometric constraint in analog design is the minimal channel length of the transistors.

### 3.1. Multi-objective problem

There are multiple objectives to address in analog synthesis. Each type of analog block has its type of specification and usually a great number of them are competitive. This means that usually improving one characteristic of the circuit decreases one or multiple others, resulting in the so called *design trade-offs*. In [23], an example of the trade-offs involved in the design of analog amplifiers is shown, such as power, gain, output swing, linearity and others.

On the other hand, optimization methods usually make use of only one cost function to measure the quality of a solution. However, on multi-objective problems one cost function for each objective is required. These cost functions can be later integrated into one through an Aggregate Objective Function (AOF) or optimized by a method that can accept multiple cost functions. This type of problem can be described by:

$$Min\ \mathbf{F}(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})\} \tag{5}$$

Subject to:

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, ..., m \tag{6}$$

$$h_j(\vec{x}) = 0 \quad j = 1, 2, ..., p \tag{7}$$

where $\vec{x} = [x_1, x_2, ...., x_n]^T$ is a vector containing the decision variables, $f_i : \Re^n \longrightarrow \Re$, $i = 1, ..., k$ are all the objective functions, and $g_i, h_j : \Re^n \longrightarrow \Re$, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions of the problem.

As there are many objectives, it is more complicated to compare different solutions than it is in single-objective problems. If one solution is better than the other in one objective but worse

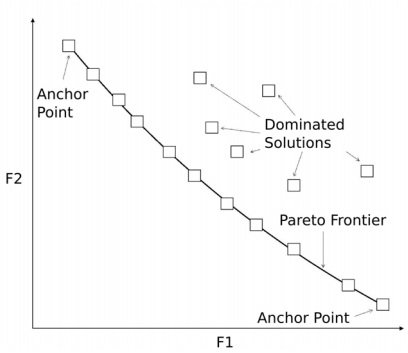


**Figure 3.** Example of a pareto front for a 2-dimensional design space.

in another, one can't choose which one is the better. However, there are cases in which one solution outperforms another in all specifications, which means one solution dominates the other.

The pareto set is the set of all non-dominated solutions. They represent all the trade-offs involved in the synthesis. Some definitions are used in [24] to formalize the concept of pareto front:

**Definition 1**: Given two vectors $\vec{x}, \vec{y} \in \Re^k$ we say that $\vec{x} \leq \vec{y}$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that $\vec{x}$ dominates $\vec{y}$ if $\vec{x} \leq \vec{y}$ and $\vec{x} \neq \vec{y}$.

**Definition 2**: We say that a vector of decision variables $\vec{x} \in \chi \subset \Re^n$ is non-dominated with respect to $\chi$, if there is not another $\vec{x}' \in \chi$ such that $\vec{f}(\vec{x}')$ that dominates $\vec{f}(\vec{x})$.

**Definition 3**: We say that a vector of decision variables $\vec{x}^* \in F \subset \Re^n$ ($F$ is the feasible region) is the Pareto-optimal if it is non-dominated with respect to $F$.

**Definition 4**: The Pareto Set $P^*$ is defined by:

$$P^* = \{\vec{x} \in F | \vec{x}\ is\ Pareto - optimal\}$$

**Definition 5**: The Pareto Front $PF^*$ is defined by:

$$PF^* = \{\vec{f}(\vec{x}) \in \Re^k | \vec{x} \in P^*\}$$

An illustration of a pareto front for a two-dimensional problem can be seen in figure 3. In this figure, both objectives are of minimization, which means a point close to (0,0) would be preferable if it was possible. The anchor points are the extreme optimal results of the specifications, while the other points in the pareto front are usually trade-offs between several specifications.

### 3.2. Continuous design space

Simulated Annealing was initially proposed to solve combinatorial problems which use discrete variables. Therefore, the generation of a new solution was accomplished by simply selecting one variable to modify and increasing or decreasing one step in the discrete space.

However, transistor, resistor and capacitor dimensions are usually better treated as continuous variables, as they can assume a large range of values. In problems with continuous variables, the minimum variation $\Delta X$ of a variable $X$ becomes subjective and at the same time fundamental to the precision and speed of the annealing process. Choosing a too much small $\Delta X$ will result in an excessive number of iterations until the algorithm finds a minimum. On the other hand, a too much great value will decrease the precision in which the design space is explored and therefore it will be difficult to gradually find the best solution.

Another issue is that different types of variables may need different variations. The minimum change in resistor's resistance $\Delta R$ which will best satisfy the speed and precision criteria is different than the minimum change in transistor's channel length $\Delta W$. The same type of variable in different types of technologies may also require the same distinction. For example, the $\Delta L$ in the channel length of a transistor in $0.18\mu m$ transistor which will be better for finding results is different from the $\Delta L$ that will achieve the same results for a $0.5\mu m$ transistor.

Other differentiation that needs to be made is based on the optimization phase. The optimization may be divided in an exploration phase and a refinement phase (which may be mixed). If a variable is far from its optimal final value, the exploration phase is more appropriate to test new values. When the variable has its value near the optimal final value, the refinement phase is better to approach the optimal solution. Modifications magnitude should be greater in the exploration phase than in the refinement phase.

Therefore, it becomes clear the need to change the variable values in an adaptive approach. The approach used for the simulations in this chapter is described in the Simulated Annealing section.

### 3.3. Generation of cost functions

The construction of a cost function for each specification is very important as it is a way to analyze how close the optimization is from the objective and also to compare which objectives should be prioritized in each stage of the optimization.

When approaching the problem through a single-objective optimizer, the several objectives are aggregated in only one using a weighted sum of each cost function as in eq. (8).

$$C = \sum_{i=1}^{n} c_i \cdot w_i \tag{8}$$

where $C$ is the total cost, $i$ is the objective index, $c_i$ is a single-objective cost and $w_i$ is the weight of the single-objective cost.

The optimizer will sense the variations on the total cost and decide if a solution must be kept or discarded. Therefore, the cost functions that compose this AOF must be designed in a way to have a greater $\Delta c_i$ for results that are far from the specifications and therefore must be optimized earlier. This greater gradient for measurements distant from the objective (e.g. less than 70% of the specification in an objective maximization) helps the optimizer to develop the circuit as a whole.

Although several types of functions can be used to create the cost function, piecewise linear functions were used in this work in order to precisely control the regions in which a greater $\Delta c_i / \Delta m_i$ should be applied, where $m_i$ is the measurement. The objective of each specification in analog design can be minimization, maximization or approximation:

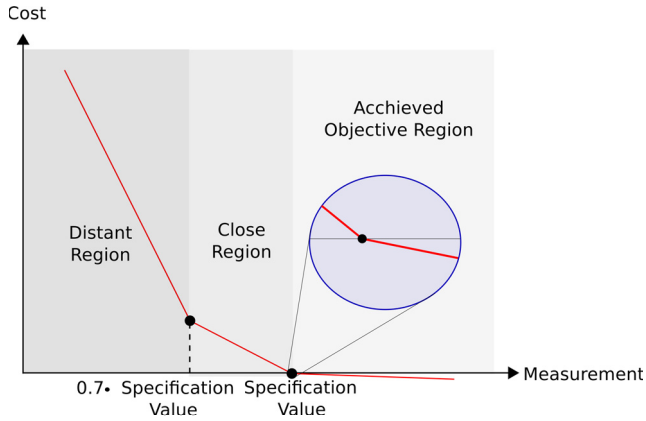


**Figure 4.** Cost function for a specification with objective maximization.

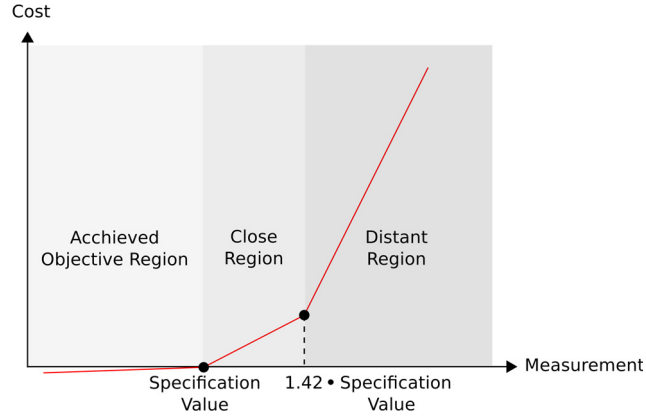


**Figure 5.** Cost function for a specification with objective minimization.

- **Maximization:** solution measurements must be equal or greater than the specification value. Examples are DC Gain, Cut-off frequency and Slew rate. Figure 4 shows how this work implements this type of function.
- **Minimization:** solution measurements must be equal or less than the specification value. Examples are area, power consumption and noise. Figure 5 shows how this work implements this type of function.
- **Approximation:** solution must approach the specification value. Examples are the central frequency of an voltage controlled oscillator and the voltage output of an voltage reference block. Figure 6 shows how this work implements this type of function.

An indicator containing the number of measures that resulted in failure is used to assign a high cost to these functions in order to provide a gradient even when dealing with non-working designs. Also, the designer can choose to put some or all transistors in a specific operation region. These and other constraints are added to the AOF in order to convert the problem from constrained to unconstrained.

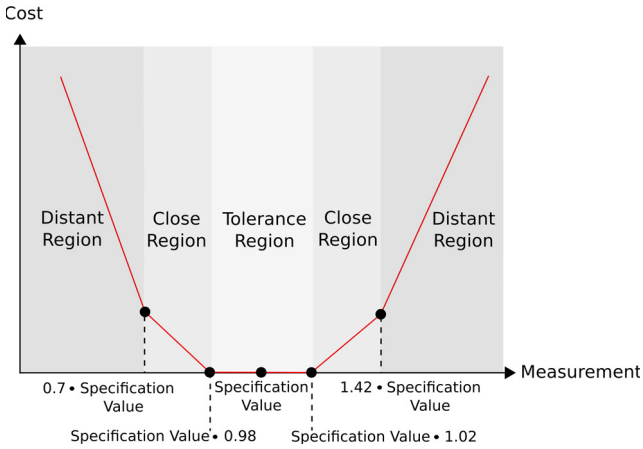


**Figure 6.** Cost function for a specification with an objective approximation.

Conversion of functional constraints into penalties is performed by adding new terms to the Aggregate Objective Function. Equation (9) shows this conversion:

$$C = \sum_{i=1}^{n} c_i \cdot w_i + \sum_{j=1}^{m} p_j \cdot w_j \tag{9}$$

where $j$ is the penalty index, $p_j$ is the penalty for each of the constraints and $w_j$ is the weight of each penalty and $m$ is the number of penalties. There is the choice of making the penalties more than simple guidelines by defining $w_i >> w_j$ for all objectives and penalties.

Other operation that helps Simulated Annealing to deal with analog design problem is the addition of a boundary check to avoid faulty simulations. The boundary check is performed every-time a new solution is proposed to avoid values out of the geometric constraints. These solutions usually result in simulation errors and take longer than typical ones. As they are irrelevant, the optimizer can skip these values and return to the state before the generation of the solution. The algorithm is kept in loop until a new solution is proposed within the boundaries and at each attempt to generate a wrong solution the sigma of the changed variable is reduced.

## 4. Simulated annealing approaches in analog design

Simulated Annealing was used in analog design since the early moments of analog synthesis was based on optimization approaches. Still now, it is one of the most popular algorithms to analog design however combined with other techniques.

In [9], a method for optimizing analog circuits using simulated annealing with symbolic simulation of the solutions is proposed. The optimization was accomplished through the use of a program called OPTIMAN and used as input design equations automatically derived using a tool called ISAAC [8]. This tool models the circuit in a set of analytic expressions which can be used by the optimization algorithm to fast evaluate each solution. Later, these tools were integrated into a synthesis environment called AMGIE [6] that covers the complete analog design flow, from specification to layout generation.

Simulated Annealing together with electrical simulations was given in [18]. Non-monotonic and adaptive normalized temperature schedule was used in order to provide fast decrease in temperature followed by reheatings. The temperature schedule could also change as a function of the percentage of accepted movements. This approach allows some level of independence of the optimization regarding the initial and final temperature values. The use of a technique called AWE (Asymptotic Waveform Evaluation) was proposed in the tool ASTRX [19] together with the use of SA. AWE could replace the SPICE simulations as it predicted the small signal behavior of the circuit using a reduced complexity model.

OPTOMEGA [13] performs a nominal optimization in order to search for the global optimal and then performs an optimization aiming robustness regarding process variations. The selection of one of the following optimization algorithms is allowed: Simplex, Simulated Annealing, Enhanced Simulated Annealing and SAPLEX (Simulated Annealing/simPLEX). This method uses Monte Carlo simulations to retrieve a value called Average Quality Index, which is used to evaluate the solutions.

MAELSTROM [15] uses Parallel Recombinative Simulated Annealing, which uses genetic algorithm characteristics with SA. A population of solution is created and at each generation the individuals can have their values changed through recombination with other individuals or through typical SA modification. An approach using SA in combination with evolutionary strategies (ES) was proposed in [1]. The Metropolis criterion from SA is used and the selection mechanism from ES. Recombination and mutation are used to adapt the length of the steps to generate the new solutions. The standard deviation based on which the new solution is made evolutes together with the solution.

In [26], a method to synthesize analog circuits oriented to robustness generating pareto surfaces is proposed. The motivation for generating pareto surfaces was to allow an efficient model of the trade-offs present on the circuit. The model can serve for a high-level synthesis tool as a way to fast evaluate low level blocks. The procedure consisted of a nominal optimization (combining simulated annealing with genetic algorithms) followed by a Monte Carlo approximation. The total time of synthesis considering nominal and process variation optimization takes only 5 to 6 times the nominal synthesis.

Although several other optimization methods can be used for analog problems (some of them can be seen in Table 1), the modern ones such as Simulated Annealing, Genetic Algorithms and Particle Swarm Optimization are specially adequate as they are robust and do not make assumptions about the problem characteristics.

## 5. SA/SQ algorithm and combination with population-based algorithms

In this section the basic Simulated Annealing/Simulated Quenching algorithm will be described, as well as the required modifications to work with continuous design space. Also, a version of SQ that uses Genetic Algorithm-based concepts and weight adjusting based on multi-objective cost function information. Finally, an approach interfacing SQ with Particle Swarm Optimization is explored.

### 5.1. SA/SQ algorithm

Simulated Annealing is a local search algorithm proposed in [14]. It has the ability to avoid local minimums by allowing worst solutions to be accepted given a certain probability.

| Category | Methods | Published in |
|---|---|---|
| Classical Methods | Nonlinear Programming | [16] |
| | Geometric Programming | [21] |
| | Sequential Quadratic Programming | [28] |
| | Linear Programming | [4] |
| Modern Methods | Simulated Annealing | [14] |
| | Genetic Algorithms | [11] |
| | Particle Swarm Optimization | [12] |
| | Ant Colony Optimization | [3] |

**Table 1.** Some optimization methods and their categories

The name Simulated Annealing is derived from the physical process called annealing in which the algorithm is based. This process is common in metallurgy and consists on heating a crystalline solid to a high temperature and then slowly cool it until its crystal lattice configuration is regular. Due to the high initial temperature, the atom movements follows a random pattern. As the temperature decreases, the movements are slowly reduced to only more regular structures yielding to a surface free of imperfections at the end of the process.

In the algorithm, the initial high temperature allows new generated solutions that are worst than the previous to be accepted. When the temperature decreases, the probability of accepting a worst solution is also decreased until only better solutions are allowed at the end of the process.

The first step of the algorithm is to evaluate the initial solution. Then, a loop as described below is performed.

- **Modification:** a small modification is made on the present solution in order to generate a candidate solution;
- **Evaluation:** the candidate solution is evaluated through the use of a cost function;
- **Decision:** based on the present temperature and on the cost difference between the present solution and the candidate solution, the probability of selecting a worst candidate solution is calculated and a random value is compared with it to make the decision;
- **Control parameter reduction:** the system temperature is decreased in order to reduce the probability of accepting a worst solution.

The stop criteria can be based on a final temperature value, number of iterations, final cost value, among others. There is a great number of variations of the simulated annealing algorithm [20] which aim to accelerate the convergence of the algorithm.

The new solution is selected according to the decision described by equation (10):

$$S_{k+1} = \begin{cases} N, & \text{if} \quad cost(N) < cost(S) \quad \text{or} \quad \tau \leq P(t_k, N, S_k) \\ S_k, & \text{otherwise} \end{cases} \tag{10}$$

where $k$ is the solution index, $N$ is the candidate solution, $S_{k+1}$ is the next solution, $\tau$ is an uniform distributed random value within $[0,1]$ and $P(t_k, R, S_k)$ is the probability of accepting a worst solution given a certain temperature $t^k$ and cost difference.

The probability of acceptance of a worst solution, also called acceptance criterion, used in this work is the Metropolis Criterion which can be seen in equation (11):

$$P(t^k, N, S^k) = e^{-\frac{Cost(S^k) - Cost(N)}{t^k}} \tag{11}$$

Theoretical proof of convergence to the global minimum for simulated annealing only exists when using a logarithmic temperature schedule. In order to acchieve solutions faster, new cooling schedules that do not have the proof of convergence derived from the original SA are used in many applications. Despite the lack of proof, these algorithms have the same structure as a simulated annealing algorithm and are usually better for the subset of problems to which they were created. Simulated Quenching is an approach to simulated annealing that allows a quicker cooling schedule and further increases in the temperature. In this work, a Simulated Quenching technique was used.

The control parameter reductions used in this work followed a simple exponential cooling scheme $t_{k+1} = \alpha t_k$, where $\alpha$ is a value close but smaller than one. The only variation used is when the detection of a local minimum occurs, in which the temperature is set as $t_{k+1} = \frac{t_0 + t_k}{2}$ where $t_0$ is the initial temperature.

### 5.2. Modification of variables

As mentioned in section 3.2, modification of variables on a continuous design space requires special efforts in relation to classical simulated annealing which deals with discrete variables. In this work we used an auxiliary variable attached to each optimization variable defining the direction and standard deviation of the step using a folded normal distribution. This auxiliary variable defining the step is updated after a new solution is generated based on how well that solution performed based on the previous one. Therefore, an adaptive step for each variable is implemented.

If the generated solution based on the provided step yielded a great improvement of the solution, the standard deviation is kept. Otherwise, if the generated solution resulted on a worse result, the standard deviation of that variable is decreased. At last, if the modification offered only a small benefit, the standard deviation is increased. The idea behind this standard deviation modification is that greater change probability must be given to variables that offer only reduced effect on the circuit when perturbed.

The equation for setting the value for the new variable is shown in equation (12):

$$X_i = X_i \cdot (1 + |f(0,1)| \cdot aux_i) \tag{12}$$

where $X_i$ is the variable being modified, $aux_i$ is the auxiliary variable that combined with $X_i$ determines the standard deviation associated with the variable, $f(0,1)$ is a normal distributed random variable with zero-mean and unitary standard deviation (which will be modified by the auxiliary variable). The standard deviation $\sigma_i$ will determine the probability of new

solutions to be close to the previous one. This standard deviation will be equal to $X_i \cdot aux_i$ $X_i$. If the auxiliary variable is smaller than 0.1, all new values within one sigma from the previous value for that variable will be less than 10% different of the previous. Therefore, a shift in the auxiliary variable sign and an increase in its values is performed when it acchieves such a small magnitude. This allows the algorithm to know in which direction a variable must be modified and update that at each iteration in which the variable is modified.

### 5.3. SA with genetic algorithm

Several algorithms that integrate Simulated Annealing with Genetic Algorithms have been proposed in the optimization field and in the analog synthesis field. These integrations usually apply the acceptance criteria and movements from simulated annealing together with the population-based nature of genetic algorithms. They keep several individuals performing SA and then, at some point, allow them to perform crossover among themselves.

On the proposed algorithm only one individual is used (as in general simulated annealing). The crossovers are made between this individual and the memory-based past solutions. The algorithm uses these past solutions to perform crossover with the present one in order to avoid local minimums when these are detected. This is possible due the existence of multi-objective information despite the fact SA is operating based on an Aggregate Objective Function (AOF). These informations are used to save the solutions that are better than all other in a single objective during the optimization. When using an AOF, the objectives in which the solution is performing worse usually are responsible for the local minimum. When the local minimum is achieved, the crossover together with a weight adjustment helps the algorithm to escape from the minimum and attempt new solutions.

Each of the objectives will have one saved solution associated to it. At each new evaluation, the present solution is compared with the saved ones in order to update the current best in each objective. Figure 7 shows a flow diagram of the algorithm and the most important steps are described above:

- **Saving the Anchor Solutions:** Choosing which solutions to save is an important step since they are going to be the responsible for taking the present solution away from the minimums. Anchor solutions are the ones that are responsible for the extreme points in a Pareto front. They represent the best solution already achieved for each given specification. Selecting directly these anchor points to be kept in the memory is not interesting for the flow of Simulated Annealing. A solution can be extremely good in $I_{supply}$, for instance, but perform bad in all other specification. Crossover between the present solution and such an anchor solution would result most likely in a non functioning solution. A more interesting approach is to save solutions that are not too far from the overall cost function achieved by the present solution. The saved solution must be better than the present solution in one specific objective but cannot be much worst in the overall AOF. This trade-off can be accomplished through a weighted sum of the cost for the specification and the overall cost.
- **Detecting Local Minimums:** detection of local minimums is based on the number of consecutive failures. If the SA algorithm is not able to accept a solution within a given number of attempts, it is assumed that a local minimum was achieved. As analog synthesis is a continuous problem, sometimes very small improvements are accepted however they

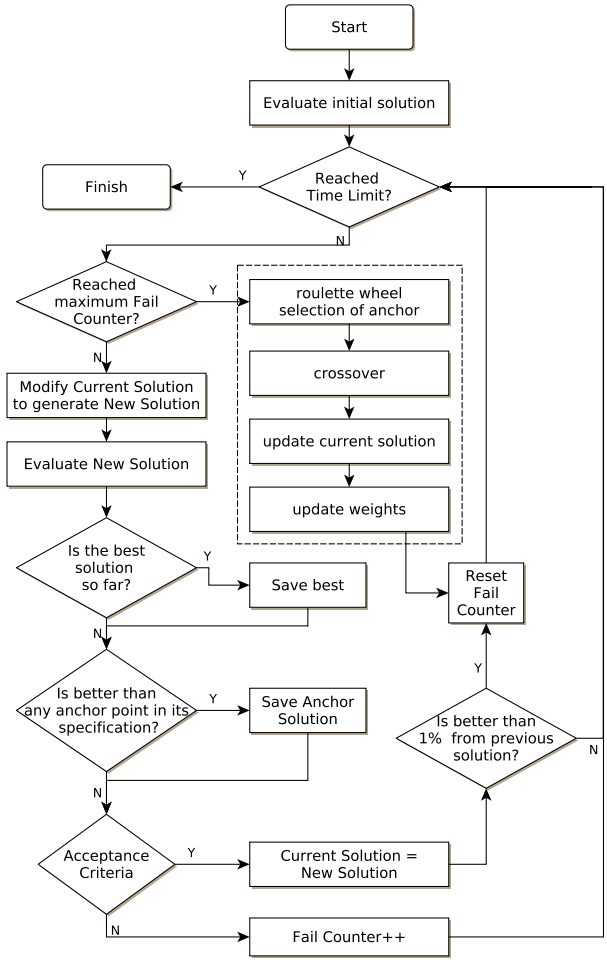


**Figure 7.** Algorithm flow for the developed SA + GA integration.

do not represent an actual significant improvement in the circuit. Therefore, it is important to reset the failure counter only when an improvement greater than 1% is accomplished.

- **Selection of anchor points to perform Crossover:** the selection of which of the previous solutions will be mixed with the present one is based on how far each of the objectives is from being achieved. The worst present objectives will grant more chance to their respective anchor solutions to be selected to crossover with the present solution. This is accomplished through a roulette selection among the anchor points.
- **Crossover Operator:** the crossovers are the responsible for taking a solution out of a local minimum. This process produces a child from two parents, which in this case are the present solution and the selected anchor solution. It is expected that the crossover between these solutions will create a child that has characteristics of the anchor (which will take the solution away from the local minimum) as well as some characteristics of the present

solution. The crossover operator used in this work is based on conditional probability and can be seen in eq. (13) and eq. (14):

$$P(g_i = present | g_{i-1} = present) = 0.8 \tag{13}$$

$$P(g_i = anchor | g_{i-1} = anchor) = 0.6 \tag{14}$$

where $g$ is the gene, $i$ is the gene index, *present* is a gene from the present solution and *anchor* is a gene from the anchor solution.

- **Weight Adjusting:** change the weights prevents Simulated Annealing from going to the same local minimum after the crossover operator was performed. This is performed by increasing the weight of the selected specification after it was sorted on the roulette wheel. It is important to keep the same scale when comparing solutions collected with different weights for the specifications, therefore the weight modifications are added only as penalties in the AOF, keeping the first terms unaltered. This allows the comparison of only the first preserved terms in order to keep the same criteria.

  Every time a local minimum is found a vector $sp$ receives a new value with specification index sorted on the roulette wheel. Equation (15) shows how the weight adjustment affects the cost after one or more local minimums are found.

$$C_A(x) = C(x) + k \sum_{i=max(1,ms-n)}^{ms} w_{sp_i} f_{sp_i} \tag{15}$$

  where $C_A(x)$ is the total cost, $k$ is a factor used to increase the weight of the adjustments on the total cost, $ms$ is the index of the last value of the $sp$ vector,$n$ is the maximum number of previous local minimums are going to be considered for the adjustment, $C(x)$ is the original total cost, $w_{sp_i}$ is the weight of specification saved on $sp_i$ and $f_{sp_i}$ is the cost value of the specification saved on $sp_i$.

### 5.4. SA with particle swarm optimization

Particle Swarm Optimization is a population-based algorithm that simulates the social behavior of birds within a flock. Its usage as an optimization algorithm was introduced by [12]. The individuals of the population are called particles and they interact with each other and the design space in search for the global minimum. The analogy of a flock of birds is the exchange of information between the particles that is converted to factors that influence their flight over the design space.

Each particle update their speed based on the following factors: inertia, experience of the particle and experience of the group. Equation (16) shows how these factors influence the speed.

$$\vec{v}_i(t) = \underbrace{w\vec{v}_i(t-1)}_{\text{inertia factor}} + \underbrace{c_1 r_1(\vec{x}_{lbesti} - \vec{x}_i(t))}_{\text{best local factor}} + \underbrace{c_2 r_2(\vec{x}_{gbesti} - \vec{x}_i(t))}_{\text{best global factor}} \tag{16}$$

where $w$, $c_1$ and $c_2$ are constants that defines the influence of each factor on the final speed, $r1$ and $r2$ are uniform random variables from 0 to 1, $\vec{v}_i$ is the velocity vector, $t$ is the time (or iteration number), $\vec{x}_{lbest}$ is the local best result and $\vec{x}_{gbest}$ is the global or neighborhood best

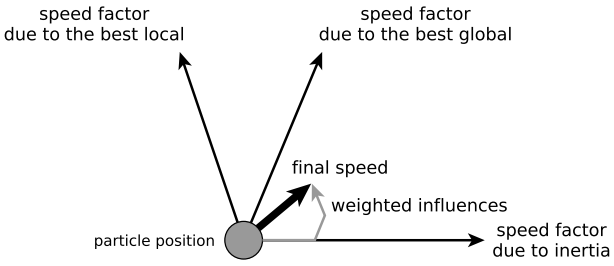


**Figure 8.** Influence of factors on the particle speed in PSO.

result. An illustration of the influence of the factors in the final speed of the particle can be seen in figure 8.

The connection topology among the particles define how a neighborhood is formed. In the simulations performed in the experiment section, the fully connected neighborhood topology was selected. In this case, all particles are in contact with the whole group so the best neighborhood solution is actually the best global solution.

In [24], a survey of multi-objective particle swarm optimization implementations is made. To treat multi-objective problems, the PSO algorithm must create and update a list of pareto solutions already visited by each particle and by the whole flock. The main goal is to maximize the number of well spread solutions in the list of pareto solutions of the flock in order to have a good approximation of the real pareto front.

For a new solution to enter on the list it must be non-dominated by any other solution already in the list. Also, at each iteration the old solutions on the list have to be checked to see if they were not dominated by new solutions. The selection of the local leader from the list of local paretos and the selection of the global leader from the list of global paretos is based on the density of solutions on the list. The leader is the one that has less close solutions already on the list. This allows the optimization to search for new solutions rather than to have a fast convergence.

Directly using PSO as circuit synthesizer in a problem with many objectives might take a long time and demand a lot of memory to keep the lists updated. Also, the required number of particles necessary to achieve a good result is very high if the particles are randomly initialized. The use of PSO for analog circuit synthesis is shown in [7] in an equation-based evaluation approach for a mono-objective and for a case with two objectives.

The proposed combination of SA and PSO tries to combine the best features of each algorithm. To accomplish that, the search for the solutions is divided in three phases. Figure 9 illustrates how this different phases are connected.

- Phase 1: Simulated annealing for multi-objective optimization using aggregate objective function. As all objectives are combined into one through weighted sum, this is a straightforward phase ideal for simulated annealing. The first objective of this phase is to find a solution that cope with the specifications and weights defined by the designer. Once this solution is found it will set all weights to the same value and start searching for a global minimum. The seed (initial solution) of this phase does not need to be good,

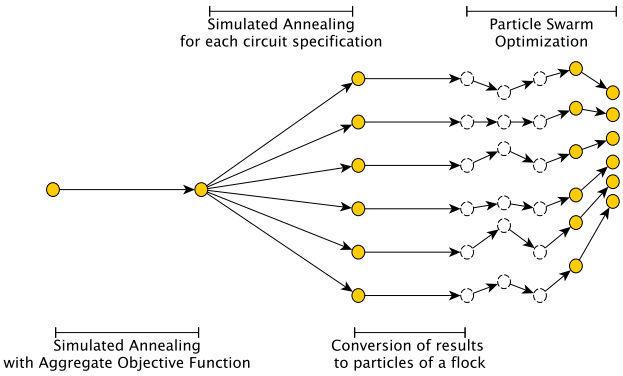


**Figure 9.** Connection between different optimization phases and methods.

although the better it is the faster this phase will end. On the example illustrated in the results section, the seed was a design with all transistor sizes set to the minimum allowed by the technology.

- Phase 2: Simulated annealing for single-objective optimization for each specification. This phase will take the best achieved circuit from phase 1 and start optimizing it for each of the specifications. All weights of the aggregate objective function are set to zero except the one of the specification being optimized. Actually, to avoid solutions that are out of the minimum specifications, only the section of the piece-wise linear function that represents the achieved specification region is set to zero. The other regions are kept for all specifications. Therefore, if a solution is outside the minimum specifications it is quickly discarded.
- Phase 3: Particle Swarm Optimization for multi-objective optimization. This phase uses the outputs of phase 2 as particles in a PSO. As each particle is already representative of an interesting part of the design space, the number of iterations to achieve an useful representation of the pareto front is reduced. The end flag for this phase is set when a sufficiently number (defined by the designer) of values from the pareto front is on the list of best globals or after a timeout.

After all phases are ended the designer still needs to have a feedback about the pareto front. However, as there might be more than two specifications, the pareto front becomes impossible to be plotted. In this work, one alternative to solve this problem was to allow the designer to select two specifications as fixed axis and plot multiple graphs of the pareto front using the rest of the specifications as the third axis of each graph. Before plotting the pareto front for each graph, a filter procedure is done to select from the n-dimensional pareto solutions which ones continue to be non-dominated if we consider only a the new 3-dimensional objective.

## 6. Experiments and results

This section will use the presented algorithms to synthesize practical circuits. Two different types of tests were performed. First, a comparison between standard Simulated Annealing and the modified Simulated Annealing allowing crossovers among anchor points and weight

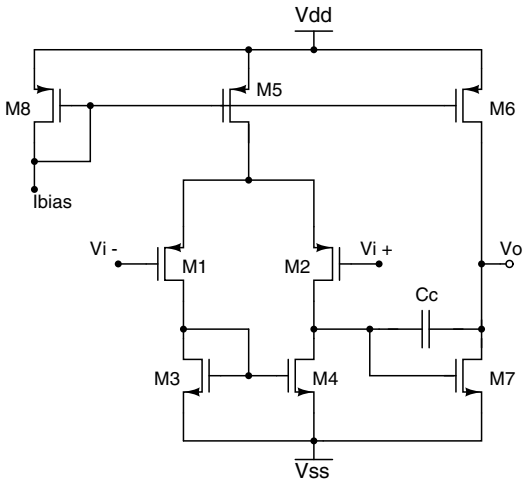


**Figure 10.** Miller amplifier topology

adjusting will be performed for two topologies. Then, a pareto front exploration using Particle Swarm Optimization together with SA will show the performance of the algorithm to illustrate the trade-offs in circuit synthesis. The computer used for these tests is a Intel(R) Core(TM)2 CPU with a 1.8 GHz frequency and with a RAM memory of 3GB.

### 6.1. Comparison between SA and modified SA with crossovers

The aim of comparing SA and Modified SA with crossovers and weight adjusting is to analyze the benefit of crossovers among anchor points in the convergence speed to a global optimum. The benchmark for this comparison is the synthesis of two amplifiers within a limited amount of time. Therefore, the optimization algorithms will not be able to continue the synthesis until their own stop criteria are met. The time limit is set to 20 minutes and then the best cost found during optimization is saved.

As the algorithms use random variables to achieve their results, several synthesis procedures are required to state a valid comparison between them. In this work, 10 synthesis for each topology in both algorithms were performed and then statistical metrics were used to compare both techniques. The metrics used are mean, standard deviation, median and also the minimal and maximum cost value for all syntheses.

The topologies used for this test are a miller amplifier (figure 10) and a complementary folded cascode amplifier (figure 11). The miller amplifier uses a voltage supply of +1.65 and -1.65 and a load capacitance of 10 $pF$ while the complementary folded cascode uses a voltage supply of +1.8 and -1.8 and a load capacitance of 20 $pF$. All the syntheses were performed using the AMS 0.35 $\mu m$ technology.

The specifications for the miller amplifier as well as the best results of all 10 synthesis found for each of the techniques can be seen in Table 2. Both techniques could achieve most of the minimal specifications within the 20-minute synthesis for at least one synthesis row. However, the statistical measurements taking into account all 10 syntheses in Table 3 show

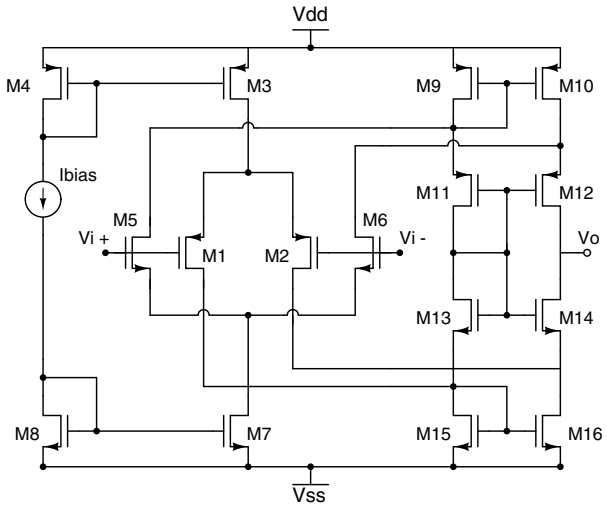


**Figure 11.** Complementary Folded Cascode amplifier

| Measurement | Specification | SA Result | SA with Crossovers Result |
|---|---|---|---|
| UGF | > 15 M [Hz] | 15.3 M [Hz] | 15 M [Hz] |
| $I_{supply}$ | < 300 $\mu$[A] | 314 $\mu$[A] | 328 $\mu$[A] |
| Phase Margin | > 60 [°] | 53 [°] | 60 [°] |
| DC Gain | > 80 dB[V] | 82.6 dB[V] | 86.0 dB[V] |
| Slew Rate (pos) | > 20 M [V/s] | 20.0 M [V/s] | 21.8 M [V/s] |
| Slew Rate (neg) | < -20 M [V/s] | -26 M [V/s] | -25.5 M [V/s] |
| ICMR | > 1.5 [V] | 2.9 [V] | 2.9 [V] |
| Output Swing | > 1.5 [V] | 2.9 [V] | 3.0 [V] |
| CMRR | > 70 dB[V] | 64.2 dB[V] | 70.7 dB[V] |
| PSRR | > 70 dB[V] | 63.5 db[V] | 70.0 dB[V] |
| Gate Area | < 1000 [$\mu$m]$^2$ | 307.4 [$\mu$m]$^2$ | 557.7 [$\mu$m]$^2$ |

**Table 2.** Specifications of the Miller amplifier and results of the best 20-minute bounded synthesis. The gray background indicates an achieved specification.

that the modified simulated annealing with crossovers performs closer to the best solutions in mean. The classical SA/SQ algorithm had solutions that were not close to acchieve all specifications and therefore had a higher final cost mean (19.9 for SA/SQ and 1.87 for SA/SQ with crossovers), representing an indicator that the crossovers can effectively improve the results. Also, it can be seen that the maximum cost result (worst result) of the classical SA/SQ is much higher than the proposed algorithm (72 for SA/SQ and 16.5 for SA/SQ with crossovers).

The same analysis was performed for the complementary folded cascode with specifications and results shown in Table 4. Similarly to the miller amplifier case, the best result achieved by each algorithm in the 10 syntheses for the complementary folded cascode are close in cost (-0.01 for SA -0.02 for SA with crossovers). However, the results from the proposed algorithm achieved a mean closer to these best results. The SA/SQ algorithm had several results far from

| Measurement | Standard SA | SA with Crossovers |
|---|---|---|
| Mean | 19.9 | 1.87 |
| Minimum | 0.20 | 0.0006 |
| Maximum | 72.0 | 16.5 |
| Standard Deviation | 29.6 | 5.2 |

**Table 3.** Statistical measurements of cost results from equation (9) for 10 syntheses limited by 20 minutes of a Miller Amplifier in 0.35 $\mu$m technology.

| Measurement | Specification | SA Result | SA+GA Result |
|---|---|---|---|
| UGF | > 10 M[Hz] | 10.3 M[Hz] | 10.1 M[Hz] |
| $I_{supply}$ | < 1 m[A] | 0.84 m[A] | 0.92 m[A] |
| Phase Margin | > 90 [°] | 87.9 [°] | 92.45 [°] |
| DC Gain | > 50 dB[V] | 60.9 dB[V] | 56.2 dB[V] |
| Slew Rate (pos) | > 13 M [V/s] | 16.4 M [V/s] | 16.6 M [V/s] |
| Slew Rate (neg) | < -13 M [V/s] | -27.1 M [V/s] | -27.7 M [V/s] |
| ICMR | > 1.5 [V] | 2.5 | 2.9 |
| Output Swing | > 1.5 [V] | 1.5 [V] | 1.8 [V] |
| CMRR | > 80 dB[V] | 91.1 dB[V] | 84.56 dB[V] |
| PSRR | > 80 dB[V] | 56.5 db[V] | 84.8 dB[V] |
| Gate Area | < 1000 [$\mu$m]$^2$ | 210.0 [$\mu$m]$^2$ | 527.0[$\mu$m]$^2$ |

**Table 4.** Specifications of the Complementary Folded Cascode amplifier and results of the best 20-minute bounded synthesis.

| Measurement | Standard SA | SA with Crossovers |
|---|---|---|
| Mean | 26.7 | 3.55 |
| Minimum | -0.01 | -0.02 |
| Maximum | 93.2 | 29.7 |
| Standard Deviation | 43.2 | 9.4 |

**Table 5.** Statistical measurements of cost result from equation (9) for 10 syntheses limited by 20 minutes of a Complementary Folded Cascode Amplifier in 0.35 $\mu$m technology.

achieving the specifications due to local minimuns and the lack of efficient ways of escaping them. This resulted in a difference of the cost mean between the techniques: 26.7 for SA/SQ and 3.55 for SA/SQ with crossovers (results are summarized in Table 5). It is important to perceive that these cost values are in a non-linear scale as the cost functions are piece-wise linear functions.

As this 20-minute synthesis benchmark is limited by time and uses the same computer hardware available for both algorithms, the mean of the results can be used to directly compare the algorithms. In both synthesis cases, the use of crossovers could improve the results effectively. Choosing this algorithm as core optimizer, the next subsection will use it together with Particle Swarm Optimization to explore the Pareto Front.

### 6.2. Pareto front exploration through modified SA with crossovers + PSO

Using the modified SA as an initial synthesis stage followed by Particle Swarm Optimization (as described in subsection 5.4), this benchmark explores the Pareto Front after trying to achieve the basic specifications. This type of synthesis is useful to highlight the design

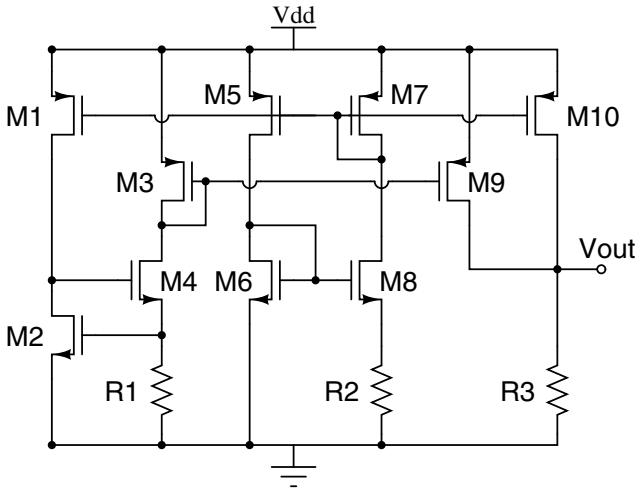


**Figure 12.** Voltage Reference Topology [17].

trade-offs. In the end of the procedure, graphs showing where the best achieved solutions are in the specification space. The topology used for this synthesis is a voltage reference as described in figure 12 [17] with a 1.25 voltage supply. Resistor values were acchieved using width and lenght dimensions. The used $0.35\mu$m technology has a high resistive module with a 1.2 k$\Omega/\square$ sheet resistance ($R_{sr}$). The final resistance is approximatedly $R = R_{sr} \cdot \frac{L}{W}$.

The specifications to be achieved and the results of the global search can be seen in Table 6. The dimensions of the devices achieved after the phase 1 of synthesis was performed can be seen in Table 7. Figure 13 shows the pareto front ploted after the Particle Swarm Optimization and the pareto filters for 3 dimensions were used. These plots were obtained by fixing two axes (Power and Area) and using the remaining specifications (TC, Noise, LR and PSSR) as the third axis of each graph.

The graphs show the relations among these specifications found on the best achieved solutions. It can be seen, for instance, that solutions with greater area and power had smaller LR and output noise than the others. As these graphs show only solutions within

| Measurement | Specification | Result |
|---|---|---|
| Power [$\mu$ W] | $< 4$ | 2.95 |
| Voltage at 27°C [mA] | 170 | 170 |
| Temperature Compensation [ppm/°C] | $< 10$ | 2.24 |
| Output Noise [$\mu$V/Hz] | $< 5$ | 1.26 |
| Line Regulation [mV/V] | $< 5$ | 4.07 |
| PSSR dB[V] | $> 40$ | 42.3 |
| Gate Area [mm]$^2$ | $< 0.01$ | 0.0096 |

**Table 6.** Specifications of the Voltage Reference and Results after global search.

| Device | W [$\mu$m] | L [$\mu$m] |
|---|---|---|
| $M_1$ | 16.8 | 10.7 |
| $M_2$ | 392 | 3.23 |
| $M_3$ | 112 | 3.93 |
| $M_4$ | 4.65 | 4.37 |
| $M_5$ | 50 | 10.7 |
| $M_6$ | 9.51 | 1.49 |
| $M_7$ | 50 | 10.7 |
| $M_8$ | 225 | 1.49 |
| $M_9$ | 11.2 | 3.93 |
| $M_{10}$ | 19.3 | 10.7 |
| $R1$ | 3.4 | 625 |
| $R2$ | 3.76 | 684 |
| $R3$ | 1.7 | 796 |

**Table 7.** Dimensions of MOS transistors and resistors of the Voltage Reference Circuit.

the minimum specifications, it can be seen in the first subplot that solutions with less power than 2.85$\mu$W usually go out of the desired specifications. However, it can also be seen that if the area is greater than 9400 $\mu m$, there are results within specifications but with high TC despite having lower power. It is important to notice that these results do not represent all the

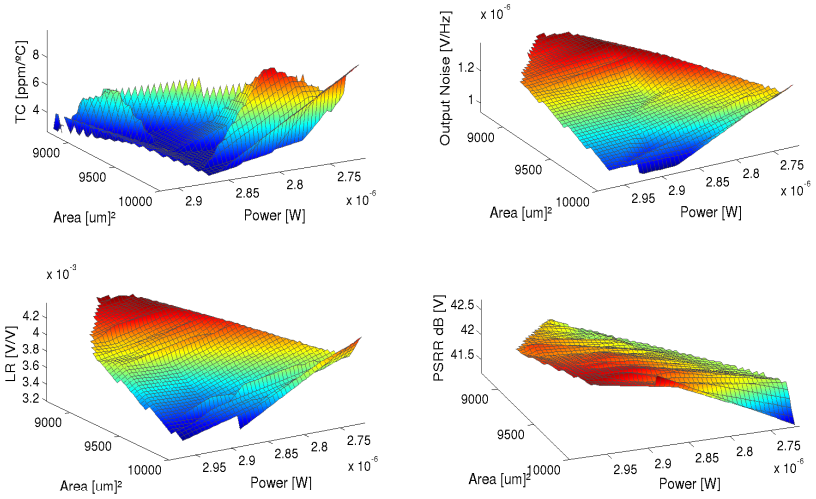


**Figure 13.** Pareto Front achieved after Simulated Annealing plus Particle Swarm Optimization

pareto solutions as the method is non-exaustive. The objective is to provide sufficient data to alow the user to make design choices.

The time required for phase 1 (global search) was 78 minutes. The time for all optimizations in phase 2 was 15 minutes and the time for the particle swarm optimization was bounded in 1 hour. Results have shown all specifications were met within a small amount of time. Therefore, the use of this technique offers good results for synthesizing and searching the pareto front of analog circuits.

In this experiment, only nominal simulations were performed. The designer can use this technique together with a optimization method taking in consideration mismatch variations [25] to further improve the circuit robustness.

## 7. Conclusion

This chapter presented the analog design synthesis problem and the use of Simulated Annealing/Quenching combined with other algorithms to solve it. The difficulties in synthesizing analog circuits are result of several factors, including high inter-connectivity among its variables, non-linearity and blurry separation among the different abstraction levels.

A brief history of the analog synthesis approaches which used Simulated Annealing was presented. SA/SQ has been used in analog circuits since the beginning of the optimization approach to analog synthesis, and it is still a popular algorithm as it yields good results and is easy to implement. Modern approaches use SA together with other algorithms in order to improve its results.

A set of experiments using SA/SQ with crossovers and also SA/SQ with Particle Swarm Optimizations were performed. The simulations have shown the combination of SA/SQ with other algorithms can improve the results considerably and also explore the pareto front. All tests were performed using a 0.35 $\mu$m technology. The circuits synthesized were a Miller amplifier, a Folded-cascode amplifier and a Voltage Regulator, and for each of them good optimization results were achieved.

## Author details

Tiago Oliveira Weber and Wilhelmus A. M. Van Noije
*Electronic Systems Engineering Department, Polytechnic School*
*University of São Paulo, Brazil*

## 8. References

[1] Alpaydin, G., Balkir, S. & Dunkar, G. [2003]. An evolutionary approach to automatic synthesis of high-performance analog integrated circuits, *IEEE Transactions on Evolutionary Computation* 7: 240–252.
[2] Balkir, S., Dundar, G. & Ogrenci, S. [2003]. *Analog VLSI Design Automation*, CRC Press, Abingdon.

[3] Colorni, A., Dorigo, M. & Maniezzo, V. [1991]. Distributed optimization by ant colonies, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, MIT Press, Cambridge, MA, pp. 134–142.

[4] Dantzig, G. B. [1963]. *Linear programming and extensions*, Princeton University Press, Princeton, N.J.

[5] Degrauwe, M. G. R., Nys, O., Dijkstra, E., Rijmenants, J., Bitz, S., Goffart, B. L. A. G., Vittoz, E. A., Cserveny, S., Meixenberger, C., van der Stappen, G. & Oguey, H. J. [1987]. IDAC: an interactive design tool for analog CMOS circuits, *IEEE Journal of Solid-state Circuits* 22: 1106–1116.

[6] der Plas, G. V., Debyser, G., Leyn, F., Lampaert, K., Vandenbussche, J., Gielen, G., Veselinovic, P. & Leenarts, D. [2001]. AMGIE-A synthesis environment for CMOS analog integrated circuits, *IEEE TCAD, 1037* .

[7] Fakhfakh, M., Cooren, Y., Sallem, A., Loulou, M. & Siarry, P. [2010]. Analog circuit design optimization through the particle swarm optimization technique, *Analog Integrated Cir Process* 63: 71–82.

[8] Gielen, G. G. E., Walscharts, H. C. C. & Sansen, W. M. C. [1989]. ISAAC: a symbolic simulator for analog integrated circuits, *IEEE Journal of Solid-state Circuits* 24: 1587–1597.

[9] Gielen, G. G. E., Walscharts, H. C. C. & Sansen, W. M. C. [1990]. Analog circuit design optimization based on symbolic simulation and simulated annealing, *IEEE Journal of Solid-state Circuits* 25: 707–713.

[10] Harjani, R., Rutenbar, R. A. & Carley, L. R. [1989]. OASYS: a framework for analog circuit synthesis, *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 8: 1247–1266.

[11] Holland, J. H. [1975]. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, University of Michigan Press.

[12] Kennedy, J. & Eberhart, R. C. [1995]. Particle swarm optimization, *IEEE International Conference on Neural Networks*, IEEE Service Center, pp. 1942–1948.

[13] Keramat, M. & Kielbasa, R. [1998]. OPTOMEGA: an environment for analog circuit optimization, *IEEE ISCAS* .

[14] Kirkpatrick, S., C. D. Gelatt, J. & Vecchi, M. P. [1983]. Optimization by simulated annealing, *Science* 220: 671–680.

[15] Krasnicki, M., Phelps, R., Rutenbar, R. & Carley, L. [1999]. Maelstrom: Efficient simulation-based synthesis for custom analog cells, *Proc. Design Automation Converence* .

[16] Kuhn, H. W. & Tucker, A. W. [1951]. Nonlinear programming, *Proc. Second Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press.

[17] Mateus, J., Roa, E., Hernandez, H. & Noije, W. V. [2008]. A 2.7$\mu$a sub 1-v voltage reference, *Proceedings of IEEE Symposium on Integrated Circuits and System Design* pp. 81–84.

[18] Medeiro, F., Fernández, F. V., Domínguez-Castro, R. & Rodríguez-Vázquez, A. [1994]. A statistical optimization-based approach for automated sizing of analog cells, *International Conference on Computer Aided Design*, pp. 594–597.

[19] Ochotta, E., Rubenbar, R. & Carley, L. [1996]. Synthesis of high-performance analog circuits in ASTRX/OBLX, *IEEE TCAD* .

[20] Pereira, A. I. & Fernandes, E. M. [2004]. A study of simulated annealing variants, *XXVIII Congreso de Estadística e Investigación Operativa*.

[21] R. J. Duffin, Bittner, L. [1967]. Geometric programming - theory and application, *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 47(8): 561–561.
[22] Rao, S. S. [1996]. *Engineering Optimization: Theory and Practice*, New Age Publishers.
[23] Razavi, B. [2001]. *Design of Analog CMOS Integrated Circuits*, 1 edn, McGraw-Hill, Inc., New York, NY, USA.
[24] Reyes-Sierra, M. & Coello, C. A. C. [2006]. Multi-objective particle swarm optimizers: A survey of the state-of-the-art, *International Journal of Computational Intelligence Research* 2: 287–308.
[25] Sáenz Noval, J. J., Roa Fuentes, E. F., Ayala Pabón, A. & Van Noije, W. [2010]. A methodology to improve yield in analog circuits by using geometric programming, *Proceedings of the 23rd symposium on Integrated circuits and system design*, SBCCI '10, ACM, New York, NY, USA, pp. 140–145.
[26] Tiwary, S. K., Tiwary, P. K. & Rutenbar, R. A. [2006]. Generation of yield-aware pareto surfaces for hierarchical circuit design space exploration, *Design Automation Conference*, pp. 31–36.
[27] Weber, T. O. & Noije, W. A. M. V. [2011]. Analog design synthesis method using simulated annealing and particle swarm optimization, *Proceedings of the 24th symposium on Integrated circuits and systems design*, SBCCI '11, ACM, pp. 85–90.
[28] Wilson, R. B. [1963]. *A Simplicial Algorithm for Concave Programming*, PhD thesis, Cambridge.
[29] Yu, G. & Li, P. [2011]. Hierarchical analog/mixed-signal circuit optimization under process variations and tuning, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30: 313.