Automatic Analog Design Procedure Including Statistical Analysis of Circuit Performance

Felipe Antunes Quirino and Alessandro Girardi
Federal University of Pampa - UNIPAMPA
Alegrete, RS, Brazil
Email: felipequirino.aluno@unipampa.edu.br

***Abstract*—Analog integrated circuits have a high range of applications, from interface circuits to signal processing. The traditional method of designing an analog circuit is based on trial-and-error. An alternative is to abstract the circuit as an optimization problem and to use automatic procedures for searching for an optimal solution. This work demonstrates an optimization method applied to a low-voltage bulk-driven operational transconductance amplifier, taking into account the high degree of variability caused by the manufacturing process. We propose a tool for design automation of analog integrated circuits which estimates performance with Monte Carlo simulation in order to evaluate the effect of process variability on circuit performance. Results demonstrate the viability of the proposed methodology in 130nm technology compared to a manual design.**

***Index Terms*—Analog IC design, Optimization algorithm, CAD tool.**

I. Introduction

Analog integrated circuits are very used in several applications, such as communication systems, Analog-to-Digital (A/D)/ Digital-to-Analog (D/A) converters, and interface circuits. The design flow of such circuits is different from the digital counterpart. It is required to size individually each device, such as transistors, resistors and capacitors, for achieving the desired electrical behavior. Each analog block has its own particular performance features, which turns the design automation difficult [1].

Traditional analog design methodology uses Simulation Program with Integrated Circuit Emphasis (SPICE) simulator to estimate circuit performance without having to build the physical system. With a SPICE tool, it is possible to perform electrical simulation for a given process technology. Thus, a designer can estimate the features of the designed circuit and verify whether or not it is necessary to change some parameter to improve performance [2]. Therefore, the traditional trial-and-error procedure of sizing an integrated circuit consists in manual sizing each component of the system followed by electrical simulations using SPICE tool. The experience of the designer is fundamental in this case, since the search for a sized circuit with good performance is dependent on the ability to adjust the correct design parameters.

An alternative is to abstract the problem into an optimization procedure. The circuit is modeled as a non-linear optimization problem where devices sizes are the free variables. Each feature of the circuit is simulated by a SPICE tool and a cost function evaluates the circuit performance. Intending to evaluate the cost function, the algorithm normalizes and makes the sum of each circuit features. A poorer result converges in a high cost, while the best solution converges into a small cost. From the cost function, the optimization algorithm perturbs the design parameters until the circuit performance converges to an optimal point. In the literature, it is possible to find optimization algorithms applied to this kind of problem, such as Genetic Algorithm (GA) [3] or Cuckoo Search (CS) [4], for example.

The technology used on analog ICs has a high degree of variability in their physical parameters, which affects directly the circuit performance.

The SPICE tool can estimate circuit performance variability by performing Monte Carlo simulation. This is the process of randomly selecting values within a range and applying to a function multiple times. The result is a Gaussian distribution curve that describes the output according to the variation of the input values of the function [5]. It is necessary to moderate the use of Monte Carlo simulation because it can demand a lot of computational resources [1]. Thus, it is necessary to reduce the computational time spent with Monte Carlo simulations when exploring the search space.

This work describes the implementation of an optimization algorithm for sizing a bulk-driven OTA described by [6]. For this propose, we model the analog sizing problem as an optimization problem and solve it for searching for a feasible solution. Part of this work is a spin-off of the work from [4], which has shown the application of optimization algorithms for automatic sizing an OTA. The contribution is the insertion of statistical analysis in the optimization flow for the search of optimal solutions under process variability.

II. Design Methodology

Fig. 1 illustrates the proposed process of analog design optimization. The optimization method initializes randomly the design parameters and sends the circuit to the SPICE tool for simulation. The simulated circuit is returned to the cost function that evaluates the result. If the simulation has a feasible cost, the algorithm applies a Monte Carlo simulation. The strategy of applying Monte Carlo only to feasible solutions reduces considerably the total optimization time, avoiding unnecessary Monte Carlo simulations. After a iteration, if the stop criterion is not achieved, the optimization method perturbs

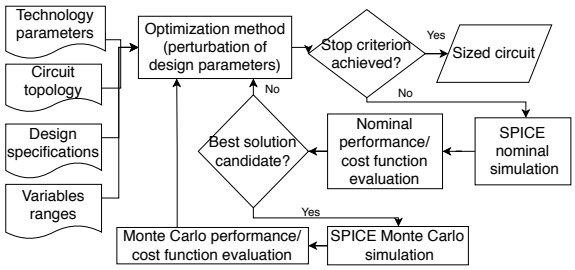

Fig. 1. Optimization procedure including statistical evaluation for best solution candidates.

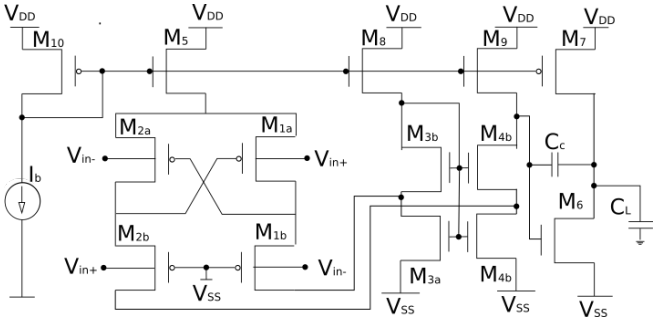

Fig. 2. Schematics of a low-voltage bulk-driven OTA [6].

the design parameters and sends the new circuit again to the simulator. The stop criterion can be defined as the maximum number of iterations or the minimal variation of the cost function, for example. At the end, the algorithm returns the best circuit found thus far.

We selected a case study for demonstrating the proposed methodology. It is based on the bulk-driven operational amplifier proposed by [6] and previously sized with a methodology proposed by [4]. The operational amplifier schematic is depicted in Fig. 2. The work of [4] used only nominal simulation for exploring the design space, thus not considering process variation. It means that the generated solution does not represent an optimum solution under process variability. Also, [4] used ideal current sources, which may not show the actual performance of the proposed system. Thus, this work intends to show the yield analysis with the complete circuit, including real current sources.

For evaluating each circuit performance feature it is necessary to execute some procedures. Two types of simulations are necessary for extracting circuit characteristics: AC simulation for extracting Low frequency Gain, Phase Margin and GBW; and transient simulation for analysing the behavior of the circuit over time, in which it is possible to extract the slew rate of the circuit.

The low voltage gain and the unity frequency gain are extracted through the circuit frequency response. Equation 1 describes how the module is calculated:

$$A_v = 20 \cdot log_{10}(Vout/Vin) \tag{1}$$

Here, $Vout$ and Vin are the circuit output and input voltages, respectively. The voltage gain is a complex function composed of a real and an imaginary part. With the module we can extract the absolute low frequency gain and with the phase we can estimate phase margin. The extraction of GBW occurs at the frequency point in which the voltage gain is unitary.

The phase margin (PM) is calculated with the following equation applied at the unity gain frequency:

$$PM = 180 - abs(angle(Vout/Vin)) \tag{2}$$

The $angle$ function gets the angle of a complex number. The abs function gets the absolute value of a real number.

Slew rate (SR) is calculated as:

$$SR = min(\frac{\Delta Vout_i}{\Delta t_i}, \frac{\Delta Vout_j}{\Delta t_j}) \tag{3}$$

where min is a function that returns the minimum element, $\Delta Vout_i$ is the variation of the output at the rising transition, Δt_i is the variation of time at the rising transition, $\Delta Vout_j$ is the variation of the output at the falling transition and Δt_j is the variation of time at the falling transition. Getting the minimum value guarantees the computation of the worst slew rate value.

The power consumption is calculated as

$$P = I_{DD} \cdot V_{DD} \tag{4}$$

Here, I_{DD} is the supply current and VDD is the supply voltage. We can separate circuit specifications in objectives and constraints. In this design we define power consumption and GBW as objective functions to be minimized (maximized). The remaining performance features are constraints in the optimization problem, such as SR, Av0, area and phase margin.

Considering that more than one feature that influences on the circuit performance, only the specification could not supply the algorithm to search for an best solution. Thus, all these specification could be abstracted as an cost function (e.g., a sum of all specifications).

The cost function must include Monte Carlo simulation for modeling circuit variation. For this, we consider all statistical results as a normal distribution, which can be uniquely described by a mean $\bar{x}_i$ and a standard deviation σ_i. Using the empirical rule, 68% of all samples are between $\bar{x}_i - \sigma_i$ and $\bar{x}_i + \sigma_i$, 95% are between $\bar{x}_i - 2\sigma_i$ and $\bar{x}_i + 2\sigma_i$, and 99.7% are between $\bar{x}_i - 3\sigma_i$ and $\bar{x}_i + 3\sigma_i$ [7]. With this rule it is possible to define Equation 5, which is the main way for evaluating the cost function in the present work.

$$f_c = \sum_{i=0}^{N} (\frac{\bar{x}_i + 3\sigma_i}{\hat{x}_i}) \tag{5}$$

The cost function is the sum of objective functions and constraint functions. The constraint function contributes nothing to the cost function if the obtained result x_i for the performance feature is greater (smaller) than a reference value $\hat{x}_i$. This reference value is defined by the user. The objective function continuously contributes positively to the cost function and

Algorithm 1 Cuckoo search via Lévy flights

Objective Function $f_c(\mathbf{r}), \mathbf{r} = (r_1, ..., r_d)$;
Generate initial population of N host nests $X_k(k = 1, 2, ..., N)$;
while (NOT stop criterion) **do**
  Obtain new solutions by Lévy flights;
  Evaluate the quality of f_{c_k};
  Choose randomly a h nest between $1, ..., N$;
  if $f_{c_k} > f_{c_h}$ **then**
    Replace h by the new solution;
  end if
  Replace a fraction (p) of the worst nests;
  Keep the best solutions of each nest;
  Sort the nests by cost;
end while

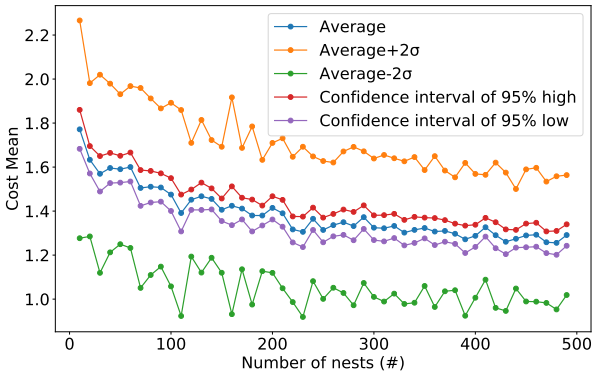

Fig. 3. Cost function behavior according to the number of nests.

decreases the value as the performance function gets smaller (larger), even if it exceeds the reference.

The optimization algorithm is responsible for choosing the design parameters since the circuit is abstracted as an optimization problem. In this work we selected the Cuckoo Search algorithm, due to its simplicity and efficiency [8].

The cuckoo search algorithm is based on the behavior of a bird specie denominated cuckoo. The cuckoo put its eggs in nests belonging to other birds. It tries to minimize the probability of the host bird discover the intruder egg by imitating its characteristics. In addition, the algorithm uses the idea of Lévy flight for exploring the design space. The Lévy flight follows a random function that tends to converge faster compared to a random flight.

Algorithm 1 demonstrates the cuckoo search algorithm via Lévy flights. Equation 6 demonstrates how we define the step of each iteration of the cuckoo search via Lévy flight.

$$\mathbf{r}_k(t+1) = \mathbf{r}_k(t) + \alpha \cdot Levy(\lambda) \tag{6}$$

Here, $r_k(t)$ is the current iteration, $r_k(t+1)$ is the next iteration and α is a scalar that defines the step between each iteration. This equation is multiplied by the Lévy function with a step of size λ.

III. Results

Cuckoo search algorithm has some parameters that must be adjusted for the given application. The most important is the number of nests N (i.e., The search number per iteration. A greater number of nests mean more simulation time. While a smaller number of nests tends in a poorer result). Intending to find the optimal number of nests for this problem, we tested the algorithm performance for different number of nests. Considering the algorithm randomness, it returns different results depending on the random seed. Thus we performed the optimization search 30 times for each number of nests. We varied the number of nests from 10 to 490. Considering that each search could take several computation hours, we used an step of 10 between the number of nests (in this case: 10, 20, ..., 490 nests). The tool designed a total of 1470 circuits with different number of nests as parameter and a fixed number of 50 iterations. The result was produced with simulations using Monte Carlo with 30 samples.

Fig. 3 shows the simulated results. Blue line represents the mean cost for each number of nests. The cost tends to decrease quickly until around 200 nests. Thus, we could select the number of 200 nests as ideal for this application. However, the average is not always the real value of every simulation. According to the empirical rule, 95% of all samples are between $\bar{x} - 2\sigma$ and $\bar{x} + 2\sigma_i$. Considering this law, the green and orange lines on Fig. 3 show the $\bar{x} - 2\sigma$, and $\bar{x} + 2\sigma$ solution behavior. The result shows that even the best solution, in the 5% of worst results, is worst than the mean of the worst solutions. Thus, in some cases, even a high number of nests can reach a bad result. It is the greater drawback of the tool. However, with a large quantity of simulations this problem is mitigated, tending to zero. Even considering the empirical rule, the mean of a sample might be reliable. According to this law, we can predict how reliable is a sample. The red and purple lines on Fig. 3 demonstrates the behavior of the 95% confidence interval in each point. The confidence interval demonstrates that the obtained mean, compared to the algorithm evolution with more nests, is close to the real mean.

Considering the behavior of the best cost according the number of nests, we selected the number of 180 nests as the best trade-off between simulation time and quality of the generated result. After this number the drop in the best average cost is practically irrelevant.

We can analyse the behavior of the algorithm for the best, median an worst solutions found for 180 nests. Fig. 4 demonstrates the evolution of the cost function for these 3 cases. For the best case, it is possible to observe that the cost decreases abruptly after 40 iterations. Comparing to the other cases, the final best case cost function is approximately 50% smaller than the median case and 1.8 times smaller than the worst case.

Table I shows each performance specifications compared to

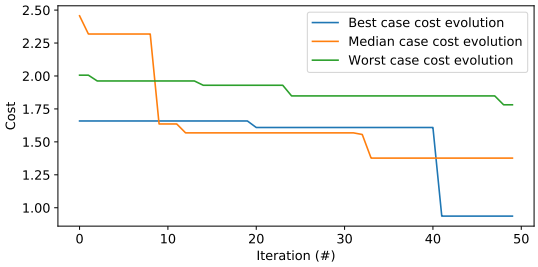

Fig. 4. Cost function evaluation.

[6], which is used as reference. The second, fourth and sixth columns shows the mean performance of the simulated circuits for the best, median and worst solutions, respectively. The third, fifth and seventh column shows the standard deviation of each specification for the same cases. Compared to [6], the best solution found in this work has better performance specifications mean in terms of GBW, PM and Power. AV0 and SR are slightly worst. However, some instances of the circuits are better in all these specifications. The median solution has a similar solution compared to the best case, however with a small GBW. No instance achieved an AV0 greater than 60dB. The worst case converged to a poor solution. With exception of GBW, this solution is worst than the reference.

Fig. 5 shows the distribution of each parameter under variability for the best, median and worst solutions, respectively. Figs. 5 (a), (b) and (c) illustrate the PM. Compared to the reference [6] (52.5) the best case has 943 instances above the reference, the median 506 and the worst solution 393. Figs. 5 (d), (e) and (f) show the GBW. for compared to [6] (1.88 kHz of GBW), the best case has 981 instances with superior performance, the median only 1 and the worst solution 995. Figs. 5 (g), (h) and (i) show AV0, which is 60dB on the reference. The best case has 70 instances with a performance superior to 60 dB and median and worst solutions have no solution above the reference. For SR (Figs. 5 (j), (k) and (l)), no sample overcame the reference of 0.77 V/ms. However, the best and the median solution achieved a SR close to this value. Power consumption (Figs. 5 (m), (n) and (o)) presented the best results, with all instances from best and median cases presenting smaller values than the reference of 18 nW. For the worst solution, only 304 instances overcame the reference. Considering all features, with exception of the of slew-rate, 53

TABLE I
PERFORMANCE INDICATORS.

	Best cost (this work)		Median cost (this work)		Worst cost (this work)		[6]
Specification	mean	σ	mean	σ	mean	σ	mean
AV0 (dB)	48.22	10.69	42.86	2.32	36.70	5.86	60.00
GBW (kHz)	3.81	0.65	1.54	0.13	2.84	0.35	1.88
PM (°)	59.99	10.15	53.68	7.56	51.59	4.78	52.50
SR (V / ms)	0.53	0.05	0.67	0.03	0.49	0.08	0.77
Power (nW)	8.86	0.22	9.57	0.24	18.19	0.45	18.00

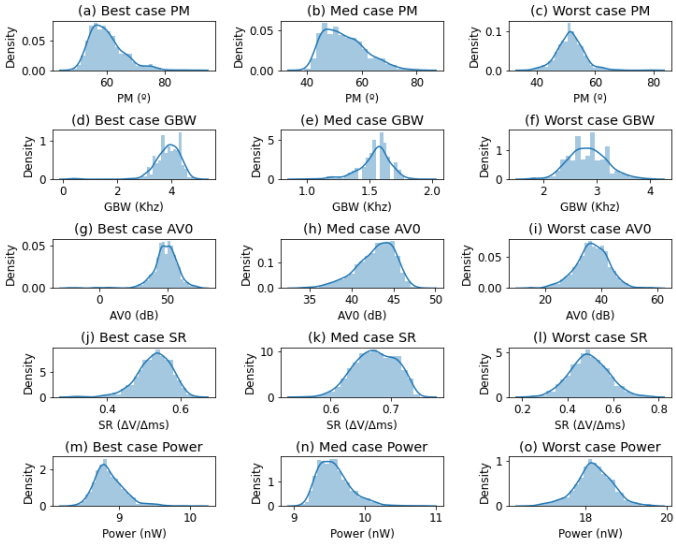

Fig. 5. Distribution of performance for generated solutions (best, median and worst cases).

instances overcame all references of [6]. Disregarding low-frequency gain, 934 instances overcame the reference on the best cases.

IV. Conclusion

This work proposed a methodology for automatic analog IC sizing under process variability. The methodology uses Monte Carlo simulation in the optimization loop to estimate circuit performance. The evaluation of the number of nests in the Cuckoo Search algorithm demonstrated that there is a trade-off between solution quality and computational time. We demonstrate that it is possible to automatically find feasible solutions with superior performance than manual design, even considering process variation.

References

[1] Helmut E Graeb. *Analog design centering and sizing*, volume 64. Springer, 2007.

[2] Paolo Antognetti and Guiseppe Massobrio. *Semiconductor device modeling with SPICE*. McGraw-Hill, Inc., 1990.

[3] Ali Jafari, Saeed Sadri, and Maryam Zekri. Design optimization of analog integrated circuits by using artificial neural networks. In *2010 International Conference of Soft Computing and Pattern Recognition*, pages 385–388. IEEE, 2010.

[4] Anderson Fortes, Luiz A da Silva, and Alessandro Girardi. Low power bulk-driven ota design optimization using cuckoo search algorithm. In *2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–7. IEEE, 2018.

[5] Kurt Binder, Dieter Heermann, Lyle Roelofs, A John Mallinckrodt, and Susan McKay. Monte carlo simulation in statistical physics. *Computers in Physics*, 7(2):156–157, 1993.

[6] Luis HC Ferreira and Sameer R Sonkusale. A 60-db gain ota operating at 0.25-v power supply in 130-nm digital cmos process. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(6):1609–1617, 2014.

[7] Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

[8] Xin-She Yang and Suash Deb. Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, pages 210–214. IEEE, 2009.