Net-Sensitivity-Based Optimization of Large-Scale Field-Programmable Analog Array (FPAA) Placement and Routing

Faik Baskaya, David V. Anderson, *Senior Member, IEEE*, and Sung Kyu Lim, *Senior Member, IEEE*

***Abstract*—Modern advances in reconfigurable technologies are allowing analog circuit designers to benefit from the computational flexibility provided by large-scale field-programmable analog arrays. With the component density of these devices, small analog circuits, as well as larger analog systems, can be synthesized and tested in a shorter time and at a lower cost, compared with the full design cycle. However, automated development platforms and computer-aided design tools for these devices are far fewer than the physical synthesis tools for their digital counterparts. One of the major reasons for this is the considerably higher impact of interconnect parasitics on circuit functionality in the analog domain; therefore, performance optimization must be recognized as an indispensable step of the analog physical synthesis flow. Our goal in this brief is to present a physical synthesis framework with an optimization core and an integrated simulation environment for verification of the synthesis results. Although SPICE has been used as the simulation tool for our experiments, there is no dependency on a particular circuit simulator. Our synthesis tool currently accepts SPICE netlists as input and gives priority to user-specified metrics when optimizing the synthesized circuit performance. Experimental results demonstrate the effectiveness of our approach.**

***Index Terms*—Analog, CAD, FPAA, reconfigurable.**

I. Introduction

ADVANCES in reconfigurable analog technologies are allowing field-programmable analog arrays (FPAAs) to dramatically grow in size, flexibility, and usefulness. With these advances, analog circuits and systems can be programmable, reconfigurable, adaptive, and implemented on standard CMOS to take advantage of scaled CMOS technology, exhibiting a density comparable to digital memories.

On the other hand, FPAAs still have not achieved the same success as field-programmable gate arrays (FPGAs) in the digital domain, even with the growing interest on and availability and use of FPAAs. This results from several factors, including lack of computer-aided design (CAD) tools, small circuit density, small bandwidth, and layout-dependent noise figures. These factors are all related to each other, making the design of a high-performance FPAA a multidimensional problem. A critical reason behind these difficulties is the nonideal programming technology, which contributes a large portion of parasitics into the sensitive analog system.

This brief is focused on making large-scale floating-gate-based FPAA technology [1] more accessible and practical. We present rapid prototyping results of several analog circuits using a floating-gate-based large-scale FPAA. A major source of parasitics introduced during the circuit mapping process is the nonideal routing interconnects. Our goal is to simulate the interconnect parasitics and minimize their impact on circuit performance. Our results indicate that we can improve the performance metrics using numerical costs that reflect the sensitivities of circuit nets to these routing parasitics.

There currently exist several CAD efforts for FPAAs in the literature [2], [3]. However, these works focus only on small-scale switch-capacitor-based designs. A survey paper [4] reviews early works in the area of programmable analog and mixed-signal circuits. Our previous work on physical mapping for large-scale FPAAs [5] provides mapping results of some analog circuits. However, the circuits used for testing are not realistic, and the method that focuses on minimizing only the switch and computational analog block (CAB) count is not capable of tracking analog performance metrics of interest to us. Therefore, it is not suitable for comparing the quality of the physical synthesis results. Our benchmark suite includes low-pass and bandpass gmC filters [6] of various orders and types.

II. Problem Formulation

A. Floating-Gate FPAA

A floating-gate element is a polysilicon layer, which is the gate of a MOSFET, with no contacts to other layers, and can maintain a permanent charge as an analog memory cell. The charge on the floating gate is modified through a combination of hot-electron injection and electron tunneling [7]. The analog components in our floating-gate based FPAA are organized in computational analog blocks (CAB); pins of these components are connected to routing wires via programmable floating-gate transistors used as switches. Details of our FPAA have been described in [1] and [5]. The routing switches in the floating-gate based FPAA connect three types of wires:

1) Type 1: Intra-CAB or vertical local wires (vlwires) connect components in the same CAB using the switches in the local crossbar. The parasitics of these wires are minimal.
2) Type 2: Inter-CAB/intra-column or vertical global wires (vgwires) connect components from different CABs

Manuscript received January 26, 2009. Current version published July 17, 2009. This work was supported in part by the National Science Foundation under Contract CNS-0411149 and Contract CNS-0347792. This paper was recommended by A. Brambilla.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: baskaya@ece.gatech.edu; dva@ece.gatech.edu; limsk@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2009.2023351

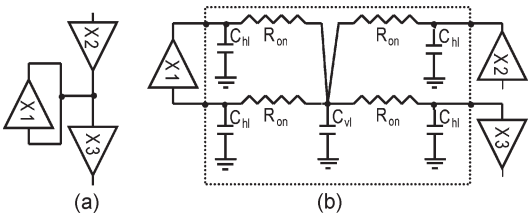

Fig. 1. (a) Four-pin net connecting three components and (b) its equivalent encapsulated interconnect circuit.

located in the same column and extend all the way from top to bottom.

3) Type 3: Intercolumn or horizontal global wires (hgwires) connect components from different CABs located in different columns and extend all the way from left to right. The parasitics of these wires are maximal due to not only the horizontal wires themselves but also the existence of additional vertical global wires on the signal path.

Some of the vgwires and hgwires are connected to the external pins of the chip. These wires are used for connecting input–output nets (ionets) to an external interface. There are also horizontal local wires (hlwires) that connect component pins to the local or global vertical wires. Since global wires vertically or horizontally span the whole FPAA without being segmented, each global wire can accommodate only one electrical connection of the circuit, which is denoted as a *net*. In addition, once a net occupies a wire, the sum of the resistance and capacitance of *all* routing switches in this wire contributes to the parasitics of the interconnect. We model the resistance of switches that are turned on $(R_{\rm on})$ and the total capacitance of all off switches on a horizontal local wire $(C_{\rm hl})$, vertical local wire $(C_{\rm vl})$, horizontal global wire $(C_{\rm hg})$, and vertical global wire $(C_{\rm vg})$. The capacitance of each wire can be computed from the number of wires intersecting that wire via switches. However, for practical devices, we can assume that $C_{\rm hl} < C_{\rm vg}$, $C_{\rm hl} < C_{\rm hg}$, and $C_{\rm vl} < C_{\rm vg}$. In case the interconnect contains more than two components (multipin net), each source-to-sink connection can be modeled individually. A sample four-pin connection net and its equivalent circuit are shown in Fig. 1.

B. Constraints, Objectives, and Challenges

The differences between analog design and digital design are reflected in the FPAA architecture and our synthesis approach. The signal integrity of an analog circuit is more difficult to maintain and can severely impact the functionality of the circuit; therefore, it is of higher priority than achieving the most compact design. To avoid adding more parasitics into the circuit, vgwires have not been segmented and have to be shared between CABs of the same column. The scarcity of global interconnect resources resulting from this architectural decision increases the challenges to the placement and routing problem.

Another challenge to be faced is the fact that parasitics not only deteriorate the performance as in digital circuits but may also completely destroy the functionality. Therefore, the impact of parasitics on performance metrics has to be monitored during the synthesis steps. A placement and routing solution that satisfies all device and net constraints may not necessarily be the desired one; the performance often has to be optimized as well. Various performance metrics may be extracted from simulation results to track the deviation from the metrics of a reference circuit. Users may give priority to the optimization of a set of these metrics over the others.

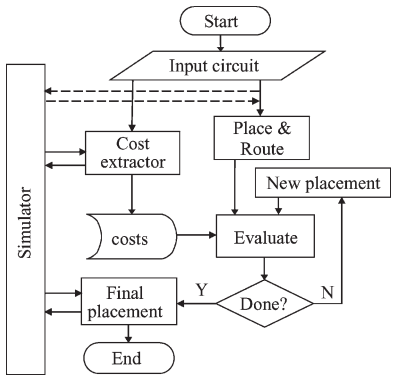

Fig. 2. Top–down prototyping framework.

The objective in FPAA placement is to pack the analog circuit components that are closely connected to each other into the same CABs. The objective in performance optimization is to obtain a placement configuration that will yield the minimum deviation from the performance of a reference circuit in terms of a user-specified set of metrics. This process is constrained by structural and design limitations. Architecture-related constraints can be divided into two groups: 1) device constraints and 2) net constraints. These constraints have been described and used in [5]. There are also design-related constraints where the user may specify the chip pin assignments to the ionets of the circuit or capacitances targeted for each net.

III. Automated Prototyping Flow

Our tool consists of the steps shown in Fig. 2. An analog circuit description is required as the first step. The circuits are described in a modified SPICE netlist, which contains additional command and design description lines, such as constraint specifications, input/output pin configurations, and optimization objectives. A SPICE netlist is an efficient way to describe analog circuit connections and also allows the use of external simulators. This netlist format also supports hierarchical circuit descriptions, as well as FPAA component library development. With this flexibility, new component definitions and their simulation models can be added to the device configuration libraries as the FPAA architecture continuously evolves.

A. Simultaneous Placement and Routing

Placement and routing for analog design differs from those for digital design. Analog systems typically consist of considerably fewer components than digital systems. This relaxes the need for a clustering phase, and the cells of a graph corresponding to the components of an analog circuit can directly be placed into their respective component positions in CABs. Because of the interconnect architecture of RASP [1], routing of a cell's corresponding nets are trivially determined with no alternatives once the cell is placed in a physical component. This transforms the routing phase into a constraint for available routing resources during the placement rather than a separate synthesis step. Therefore, placement and routing simultaneously take place in our tool. Each CAB in the FPAA is ranked for cells of the circuit graph in the order given by the modified

hyper-edge coarsening algorithm that we used for clustering in this combined placement and routing phase. The initial placement result obtained in this phase is refined in the next step, which is described in Section III-B.

B. Optimization of Circuit Performance

The interconnect model defined in Section II-B shows that the FPAA has three basic types of wires, each carrying different parasitic values into the synthesized circuit. It is possible to model routing interconnects by replacing each net between neighboring component pins with an equivalent subcircuit consisting of *RC* branches that connect its pins to each other to reflect the wire and on/off switch impedances, as shown in Fig. 1. Simulation of the circuit with and without this subcircuit allows observing the impact of routing parasitics on the circuit performance. The accuracy of simulation results depends on the sophistication of the employed models. In [9], a similar model has been used to compare the simulation results to the measurements from a synthesized circuit to show that the simulation and measurement results are well correlated. Since this model guides the direction of optimization, we are interested in the model's fidelity rather than its very high accuracy; therefore, this level of accuracy is sufficient for use by our optimization engine. Our tool can synthesize netlists for different placement and routing configurations to be simulated by SPICE and can extract metrics of interest from the simulation results. Although using an integrated simulator would increase the tool performance, employing an external simulator brings more flexibility in adopting different simulators as they improve in accuracy and speed.

While routing parasitics mostly affect the performance in the digital domain, they may have a serious impact, even on the functionality of an analog circuit. Therefore, the emphasis here is mainly placed on optimizing the performance to meet design requirements rather than achieving the most dense packing of the components. During the processes of placement and routing, each circuit net is associated with a different wire type in FPAA. An analog design expert can easily recognize which nets are more sensitive to the routing parasitics than others for a given circuit and will consider this when making decisions for assigning wire types to each net. We can automate the same behavior by simulating the circuit with different wire types assigned to each net using the interconnect model previously described and comparing the performance metrics that we are interested in. If a net is more sensitive to the routing parasitics, the metrics obtained by simulating that net assigned to different wire types will deviate more from the metrics of a reference circuit, compared with those of a less sensitive net. We extract a numerical value from this deviation to generate a net-sensitivity matrix where each entry corresponds to the cost of using a wire type for a net when routing the circuit. In our FPAA with three wire types, this net-sensitivity matrix is of size $3 * n$ for a circuit with n nets.

To extract costs for the net-sensitivity matrix, we compare the simulations of different net-wire assignments based on a reference circuit derived from the circuit to be optimized. Ideally, one would prefer to use wires with the smallest parasitics for every net, choosing a target circuit configuration as one that uses only local interconnects for every circuit net. This may not be the case for every circuit since a higher capacitance may sometimes be required for certain nets when FPAA CABs lack large drawn capacitors to satisfy the circuit's requirements. In addition, nets connected to the FPAA input–output pins are limited to certain wire types and therefore may not be assigned to the smallest capacitance wire type. Considering all these factors, a reference circuit configuration that adds the approximate effects of the nonideal interconnects while ignoring device and routing constraints is used to obtain target performance metrics that will guide the tool during circuit refinement. To find the sensitivity of a net, we modify the reference (target) circuit by changing the interconnect type associated by that net while keeping the remaining nets in the target circuit configuration and collect the metrics from the simulation results. Since one entry for each net belongs to the target circuit configuration, $2n$ different circuit configurations (two interconnect types for each of the n nets) are simulated to find $2n$ entries of the sensitivity matrix, leaving the remaining n entries zero. In this way, the whole matrix can be populated in at most $2n + 1$ simulations.

The user may specify the optimization objectives in three directions: 1) equalization; 2) maximization; and 3) minimization. Each direction uses the following formulas to compute the net-interconnect costs associated with the metrics of interest:

$$\text{Equalize} : c_{ijt} = \frac{|X_i - x_{ijt}|}{\max(1, |X_i|)}$$
$$\text{Maximize} : c_{ijt} = \frac{X_i - x_{ijt}}{\max(1, |X_i|)}$$
$$\text{Minimize} : c_{ijt} = \frac{x_{ijt} - X_i}{\max(1, |X_i|)}$$

where x_{ijt} is the value of metric i when net j is connected to wire type t, X_i is the target value for metric i, and c_{ijt} is the interconnect cost for metric i of net j connected to wire type t. If multiple design objectives are specified at the same time, the overall interconnect cost of net j for wire type t (nc_{jt}) can be found as the weighted sum of c_{ijt} for each metric i

$$nc_{jt} = \sum_{i}^{m} w_i * c_{ijt}$$

where w_i is the weight of metric i, and m is the number of metrics used for optimization. During the initial placement and refining phases, the net-interconnect cost of net j can be found by substituting the value of t, such that $nc_j = nc_{jt}(t)$.

In this work, we chose simulated annealing [10] to test the effectiveness of net-interconnect costs in an optimizer. Starting with a valid component–CAB assignment configuration, components are moved into eligible vacancies in other CABs or swapped with components of the same type to facilitate new configurations. Our objective function consists of a weighted sum of the total switch number $numsw$, net-wire configuration costs nc_j, and number of columns spanned by the net $numcol_j$, i.e.,

$$\text{cost} = \alpha * numsw + \beta * \sum_{j}^{k} (nc_j + numcol_j)$$

where k is the number of nets, and α and β are constants. Here, nc_j reflects the sensitivity of net j to the three wire types,

TABLE I
PERFORMANCE METRICS AND THEIR DEFINITIONS

metric	definition	optimization objective
fc	cut-off frequency [Hz]	equalize to target
gpass	pass-band gain [dB]	equalize to target
rp	pass-band ripple [dB]	minimize
ror	roll-off rate [dB per decade]	maximize

TABLE II
TEST BENCH CIRCUITS

filter name	filter type	order	#cmp	time(s)
b_lp8	butterworth LP	8	16	7.7
bs_lp7	bessel LP	7	16	7.4
c1_lp5	chebyshev LP	5	12	5.5
c2_lp5	inverse chebyshev LP	5	19	13.3
e_lp7	elliptic LP	7	32	34.0

whereas $numcol_j$ reflects the impact of additional columns if a type-3 wire is used ($numcol_j$ is 0 for vlwire and vgwire). It is also possible to implement other optimization approaches that exploit the computed net-interconnect costs as long as the device constraints can be adopted into the engine.

IV. EXPERIMENTAL RESULTS

A. Test Suite Setup

Testing the quality of FPAA mapping results has been a difficult issue because of the lack of a realistic circuit suite built using the available component set. Having a variety of circuit classes, each with their own relevant metrics, prevents testing from being a straightforward process. In this work, we propose a test suite of our own with focus on filters in an attempt to establish a uniform testing criteria. The availability of built-in capacitors and OTAs in FPAA CABs makes gmC filters suitable candidates for this purpose. For this test suite, we are only concerned with ac metrics that are relevant to important filter specifications. These metrics and their definitions can be found in Table I. We want to observe the impact of incorporating a performance metric into the optimization cost function for each circuit. Unlike digital circuits, it is not possible to automate the measurement of a suite of analog circuits, and the manual measurement process is prohibitive as the number of circuits and experiments increases. Therefore, we include the simulation results in this brief, relying on the correlation between the measurement and simulation results presented in [9].

A gmC filter can be built by adjusting the transconductance (gm) of the OTA using state-space equations of any desired filter specifications [11]. We use MATLAB to generate state-space equations and SPICE netlists for five types of low-pass filters, i.e., Butterworth, Chebyshev, inverse Chebyshev, Bessel, and elliptic. Although the user has the freedom to generate gmC filters of different orders and pass types, we are able to display only five representative low-pass filters of orders up to 8 due to the limited space. Filter type, order, number of components (#cmp), and average synthesis time (including simulations, place and route, and optimization for the tested metrics) for each circuit are given in Table II.

B. Synthesis Results

We ran these test circuits using our tool and WinSpice3 [12] running on a Windows XP Pentium 4 2.4-GHz machine with 1 GB of random access memory to collect the performance metrics listed in Table I for different optimization goals. These goals include rp minimization (*rp opt*), ror maximization (*ror opt*), gpass (*gpass opt*), or fc (*fc opt*) equalization to target performance metrics, and switch number minimization (*sw opt*) to be a comparison with earlier work in [5]. For each optimization goal based on a performance metric, we inspect that metric alone in net sensitivity cost extraction. We obtain the performance metrics of circuits for each of these optimization goals, along with the initial placement solution (*pre_opt*), and present them in Figs. 3–6. In Fig. 3, we demonstrate the relative error of the resulting fc value from the target fc. All obtained fc values in Fig. 3 are within the 2.5–4.5-kHz range. Since the target gpass values are close to 0 dB, we chose to demonstrate the absolute errors rather than the relative errors for gpass in Fig. 4. Figs. 5 and 6 present the rp and ror values. Since the optimization objective is maximization for ror, higher values mean success. For all other metrics, we want to see lower results or lower errors with respect to the target values.

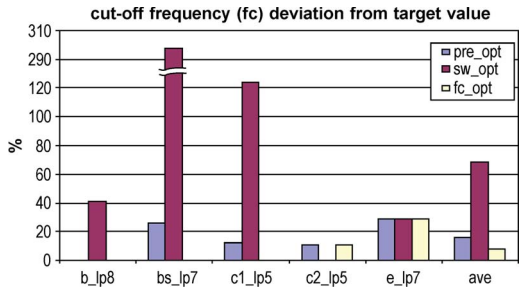

Fig. 3. Cutoff frequency (fc) errors for different optimization goals (the lower the value, the better; fc_opt gives the lowest average values). Missing bars mean zero.

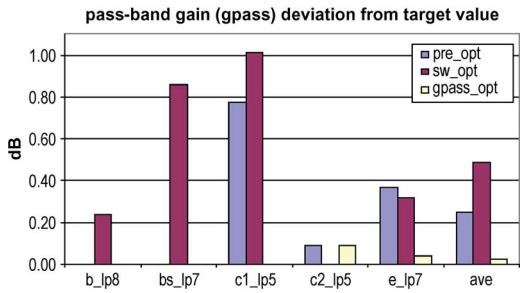

Fig. 4. Passband gain (gpass) errors for different optimization goals (the lower the value, the better; *gpass_opt* gives the lowest average values). Missing bars mean zero.

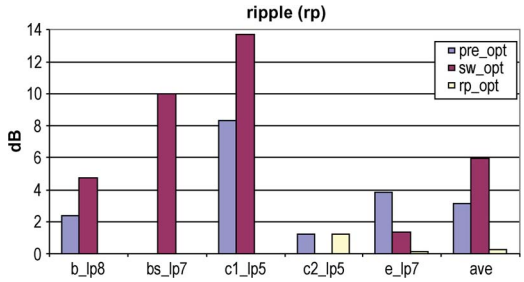

Fig. 5. Passband ripple (rp) values for different optimization goals (the lower the value, the better; *rp_opt* gives the lowest average values). Missing bars mean zero.

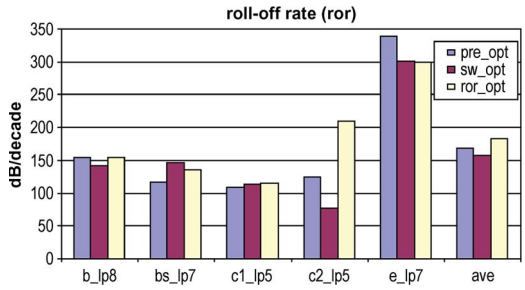

Fig. 6. Rolloff rate (ror) values for different optimization goals (the higher the value, the better; *ror_opt* gives the highest average values).

V. Analysis

Using an analog circuit simulator in an optimization loop that requires multiple iterations is prohibitively time consuming. In this work, we replace the simulator with numerical net sensitivity costs in order to complete the optimization task in a feasible time while pursuing the performance goals set by the user. In a digital circuit, reducing the critical path length, which corresponds to minimizing the number of switches in our case, would be a good approach to reduce the circuit delay. When the results presented in Figs. 3–6 are inspected, one can see that switch number minimization alone can actually degrade the circuit performance, whereas using net sensitivity costs extracted from performance metrics results in better or similar performance, compared with that of the initial circuit or switch minimization alone. Fig. 3 reveals a 50% reduction by using fc_opt, compared with pre_opt for the fc error from the target fc value. Likewise, we observe a 90% error reduction for gpass in Fig. 4, a 91% decrease in the rp value in Fig. 5, and a 9% increase in the ror value in Fig. 6 when net sensitivity costs based on these metrics are used. The only case where switch minimization seems to work slightly better is c2_lp5 (inverse Chebyshev filter), which is a robust circuit with acceptable performance metrics for all optimization methods.

In these experiments, only one performance metric is used to form the cost function for each optimization method. Simultaneously optimizing multiple metrics requires special care since some metrics will be conflicting with each other, and it will not be possible to optimize all at the same time. For instance, it is very difficult to increase the rolloff rate and decrease the ripple of a circuit at the same time. In addition, since the metrics have different numerical value ranges, their relative weights should carefully be assigned.

VI. Conclusion

In this brief, we have presented a rapid prototyping framework targeting floating-gate-based FPAA that can extract and use numerical simulation metrics. Our synthesis tool accepts SPICE netlists as input. Our goal is to get optimized performance relative to a target circuit configuration. Although analog circuit sizes are typically smaller than digital circuits, the design specifications can considerably be harder to achieve, and the parasitic effects not only have an impact on the performance but also can damage the functionality of the circuit. Therefore, one's main concern in analog design should be focusing on the solution quality rather than the switch or CAB quantity in the final placement. Experimental results show that the performance of the resulting circuit is degraded if focused only on the number of switches and not on the different net sensitivities. Our tool provides an automated method for extracting and using net sensitivities in optimization.

References

[1] T. Hall, C. Twigg, P. Hasler, and D. Anderson, "Developing large-scale field-programmable analog arrays," in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, 2004, p. 142.

[2] H. Wang and S. Vrudhula, "Behavioral synthesis of field programmable analog array circuits," *ACM Trans. Design Autom. Electron. Syst. (TODAES)*, vol. 7, no. 4, pp. 563–604, Oct. 2002.

[3] S. Ganesan and R. Vemuri, "Behavioral partitioning in the synthesis of mixed analog-digital systems," in *Proc. Des. Autom. Conf.*, 2001, pp. 133–138.

[4] D. D'Mello and P. Gulak, "Design approaches to field-programmable analog integrated circuits," *Field-Program. Analog Arrays*, vol. 17, pp. 7–34, 1998.

[5] F. Baskaya, S. Reddy, S. Lim, and D. Anderson, "Placement for large-scale floating-gate field-programmable analog arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 8, pp. 906–910, Aug. 2006.

[6] P. Allen and D. Holberg, *CMOS Analog Circuit Design*. London, U.K.: Oxford Univ. Press, 2002.

[7] P. Hasler, C. Diorio, B. Minch, and C. Mead, "Single transistor learning synapses," in *Adv. Neural Inf. Process. Syst.*, 1995, pp. 817–826.

[8] J. Gray, C. Twigg, D. Abramson, and P. Hasler, "Characteristics and programming of floating-gate pFET switches in an FPAA crossbar network," in *Proc. IEEE ISCAS*, 2005, pp. 468–471.

[9] F. Baskaya, B. Gestner, C. Twigg, S. Lim, D. Anderson, and P. Hasler, "Rapid prototyping of large-scale analog circuits with field programmable analog array," in *Proc. 15th Annu. IEEE Symp. FCCM*, 2007, pp. 319–320.

[10] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[11] T. Laxminidhi and S. Pavan, "Design centering high-frequency integrated continuous-time filters," in *Proc. IEEE ISCAS*, 2007, pp. 1939–1942.

[12] M. Smith, WinSpice. [Online]. Available: http://www.winspice.co.uk/