Architectural and Parametric Optimization of Low-Pass RF Analog Filters in VHDL-AMS Based High-Level Synthesis

T. J. Kazmierski and F. A. Hamid
School of Electronics and Computer Science, University of Southampton, UK
tjk@ecs.soton.ac.uk, fah99r@ecs.soton.ac.uk

ABSTRACT

This contribution presents architectural and parametric optimization techniques developed and implemented for FIST, an automated high-level synthesis system of analog filters based on behavioral VHDL-AMS descriptions. The application of FIST and its optimizer to low pass RF filters are discussed and illustrated with a case study of 1GHz Butterworth filter design. Performance evaluation is based on a combination of accuracy and power consumption. The candidate architectures selected and evaluated by the synthesizer are presented their performance characteristics are discussed.

1. INTRODUCTION

High-level automated synthesis is broadly defined as a process of converting abstract specifications of hardware into optimized structural implementations. Typical stages of high-level synthesis involve exploration of possible architectures, parameter and constraint exploration for candidate architectures, performance model generation and performance evaluation. As complex System-on-Chip (SoC) designs with a mixture of complex analog and digital blocks are becoming increasingly more common, it will be practically impossible to benefit from future technological advances without the use of suitable high-level synthesis tools. Although often representing a small part of a mixed-signal SoC, the analog part often causes the bottleneck in design time and effort. This is because analog design automation is still lagging behind its digital counterpart. Therefore, particular attention needs to be devoted to new high-level synthesis methodologies for mixed-signal and analog specifications. A number of approaches have recently been proposed. One of the earliest methodologies is the one implemented in the VASE environment (VHDL-AMS Synthesis Environment) [1-3], which performs analog synthesis from functional specifications in VHDLAMS. That approach, as well as many others [4-6], start from graph-like specifications or parse trees derived from the description [7], produce various architectures that suit the specification, and then optimize their parameters. The results of each synthesis step are passed to the next step after non-viable solutions are rejected. It has to be noted that while earlier analog synthesis tools are mainly knowledge-based [8-10], in which synthesis mainly relies on a database of information regarding circuit structure and behavior, the more recent approaches are optimization-based [11-13], where the synthesis problem is viewed as an optimization task. For the synthesis of analog circuits, there are several basic tasks common in the design and building of an analog module: (1) specification, (2) topology generation, (3) performance evaluation and (4) structure generation. This contribution concentrates on the three latter aspects, which heavily rely on optimization, and our specific application is targeted towards the synthesis of continuous-time low-pass RF filter circuits. An automated synthesis environment, named FIST [14], has been developed for this purpose. The input of FIST is a high-level VHDL-AMS behavioral description and the output is a HSPICE netlist of the best filter topology that realizes the input specifications. FIST recognizes and hence, can synthesize, certain types of filter elements either in the time domain using differential-algebraic equations (DAEs) or in the frequency domain using s-domain transfer functions.

2. OUTLINE OF SYNTHESIS ALGORITHM

Figure 1 shows the implementation of our synthesis methodology for analog filters. The process begins with a scan of the VHDL-AMS parse tree for synthesizable syntax to finally produce a filter netlist that will be parametrically optimized using a three-tier optimization system. The three tiers of the optimization process comprise: (1) stochastic search, (2) non-linear simplex and (3) the built-in HSPICE curve-fitting optimizer.

When the tree scanner recognizes a synthesizable construct, it employs a static calculator to search the parse tree in a recursive manner, to evaluate the required filter coefficients from the user defined VHDL-AMS code. This process extracts the filter specification. High order filters (i.e. of order greater than 2) are mapped into configurations of first and/or second order cells. The resulting configurations are then parametrically optimized and the best configuration is selected. The parametric optimization relies on the error figure calculated as the difference between the desired and actual AC characteristics. AC analysis is performed by full HSPICE simulation around a DC operating point whose evaluation is included in the optimization.

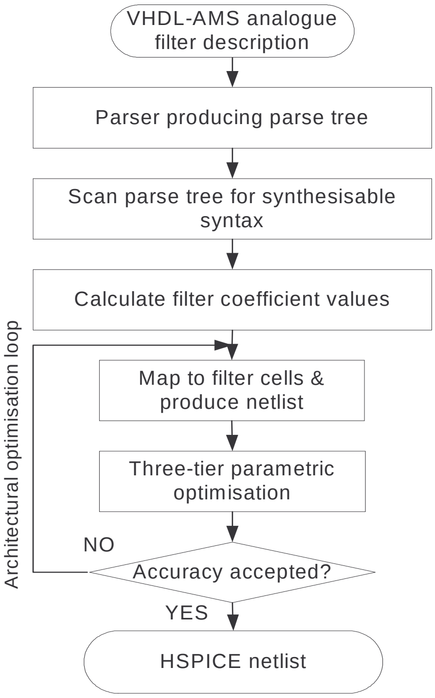
VHDL-AMS analogue filter description
Parser producing parse tree
Scan parse tree for synthesisable syntax
Calculate filter coefficient values
Map to filter cells & produce netlist
Three-tier parametric optimisation
Accuracy accepted?
NO
YES
HSPICE netlist
Architectural optimisation loop

Figure 1: Flow chart of FIST, the VHDL-AMS architectural synthesis system for high-frequency analog filters.

FIST uses a filter cell library which is a collection of HSPICE parametrized filter netlists with topologies suitable for high-frequency integrated-circuit implementation. The main types of cells used by FIST for low-pass filters are based on OTA-C (Operational Transconductance Amplifier-Capacitor) pass biquads and all-pole multiple loop feedback structures of the LF (Leapfrog Feedback) and IFLF (Inverse Follow the Leader Feedback) types.. Each filter cell in the library serves as a template where the critical parameters that influence and affect the filter performance and specification are selected for optimization. The main performance measure for each filter cell is the accuracy of the match between the simulated frequency response and the ideal curve obtained from the VHDL-AMS specification. Other performance criteria are power consumption and area. This technique has proved particularly useful for high-frequency filters in which parasitic effects and high-order MOS effects such as sub-threshold and very short channel behavior cannot be ignored. One important point regarding the effect of parasitic capacitances is that the effective MOS transconductance itself becomes frequency-dependent, and it can drop significantly with the increase in frequency. This is particularly important in the optimization of OTA filter cells which depends largely on the optimization of the specific OTA cell alone.

3. ARCHITECTURAL OPTIMIZATION

Architectural optimization is performed by analyzing various filter topologies, parametrically optimizing them using the selected performance criteria and finally choosing the best one. Three key aspects must be addressed: 1) parametric optimization engine, 2) evaluation method, and 3) performance model. In our case the performance model is based on a weighted combination of the frequency response accuracy and power consumption. The strategy is illustrated in Figure 2, where the performance model is the AC response data. The optimization engine is based on a three-tier optimizer as described below. The evaluation method relies on full HSPICE AC analysis performed for each candidate topology within each iteration of the innermost optimization loop. The optimizer returns the cost function value for each topology to the architecture evaluator, which selects the lowest cost function topology
Filter cells are obtained from a collection of HSPICE netlists of analog filter topologies suitable for high-frequency applications. To increase the optimization accuracy, and likelihood of finding the global minimum, the parametric optimizer employs a three-tier optimization strategy (Figure 3), which is a combination of stochastic search, downhill simplex algorithm [15] and HSPICE's built-in optimization engine [16]. The downhill simplex, or also known as the amoeba, is based around manipulation of a multi-dimensional simplex. The built-in optimization engine in HSPICE uses a combination of the steepest descent and Gauss-Newton methods as proposed by Levenberg and Marquardt [17]. The steepest descent algorithm is applied when the current search is far from the solution, hence large and rapid movements are made to find a better point. When the algorithm detects that the search is improving too slowly, the Gauss-Newton method is used. We have found that the HSPICE optimization algorithm is very sensitive to the starting point and frequently does not converge to a good local minimum. Unfortunately cost functions calculated from simulation results of analog circuits tend to have many local minima. Hence, the FIST optimizer relies on stochastic search to generate good starting points as shown in Figure 3.

4. PERFORMANCE EVALUATION

The accuracy measure is obtained from the error figure generated by the HSPICE curve-fitting algorithm, which runs in the innermost loop as shown in Figure 3. It realizes the following least-squares estimate of how much the simulated AC frequency response differs from the magnitude of the desired response [16]:

$$err_{total} = [\frac{1}{n} \cdot \sum_{i=1}^{n} err_i^{\,2}]^{1/2} \qquad (1)$$

where,

$$err_i = \frac{M_i - C_i}{\max(MINVAL, M_i)}, i = 1, n \qquad (2)$$

where *n* is the number of frequency points, *M* and *C* are the measured and calculated values respectively, and *MINVAL* is defaulted at 1e-12 to prevent floating-point errors.

The multi-objective optimization for accuracy and power cannot be done by HSPICE optimization alone, as this would involve different types of HSPICE analyses – AC analysis for frequency response, and operating point or transient analysis for power - using the same set of parameters. As information regarding power consumption of the candidate can be obtained from HSPICE optimization results, it is possible to find a set of solutions which are both accurate and exhibit reasonably low power consumption for a particular topology. Therefore, the formulation of the total cost function *CF* that guides the outer optimization loops can be given as follows:

$$CF = err_{total} \cdot w + Power + err_{bp} \qquad (3)$$

where *w* is a weighting factor to ensure that a penalty given to err_{total} is high. *Power* is the total power consumption in Watts, and err_{bp} is a penalty calculation specific to the band pass filter only. The stochastic search serves primarily as a tool to explore the local minima present in the surface of analog circuit optimization cost function. The initial points generated stochastically are used by the simplex-based optimization, which moves the initial points towards a solution that optimizes the accuracy and power of the candidate design. The downhill simplex algorithm uses HSPICE's optimization to obtain the set of parameter values that gives the closest fit to the specified frequency response curve.

The values of the weighting factor *w* are selected to give more prominence to the accuracy, and effectively ignore the power consumption, if the accuracy errors are excessively large (Table 1).

Curve-fitting error value	*w*
$err_{total} > 0.9$	10
$0.6 < err_{total} \le 0.9$	5
$0.4 < err_{total} \le 0.6$	4
$0.3 < err_{total} \le 0.4$	3
$0.2 < err_{total} \le 0.3$	2
$err_{total} \le 0.2$	1

Table 1: The penalty values assigned to the curve fitting optimization cost function given by eqn (3).

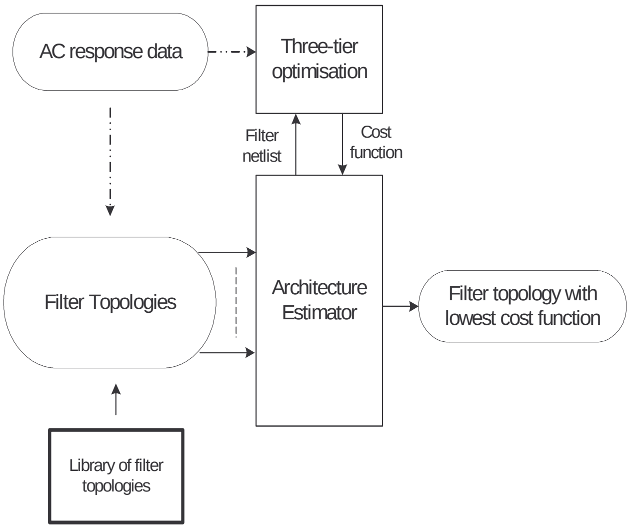

Figure 2. Architectural optimization process in FIST.

4. CASE STUDY: SYNTHESIS OF A 1GHZ 4TH LOW PASS FILTER

This case study presents the synthesis of a fourth-order lowpass Butterworth filter with a cut-off frequency of 1GHz. The VHDL-AMS specification for this filter is presented in Figure 4.

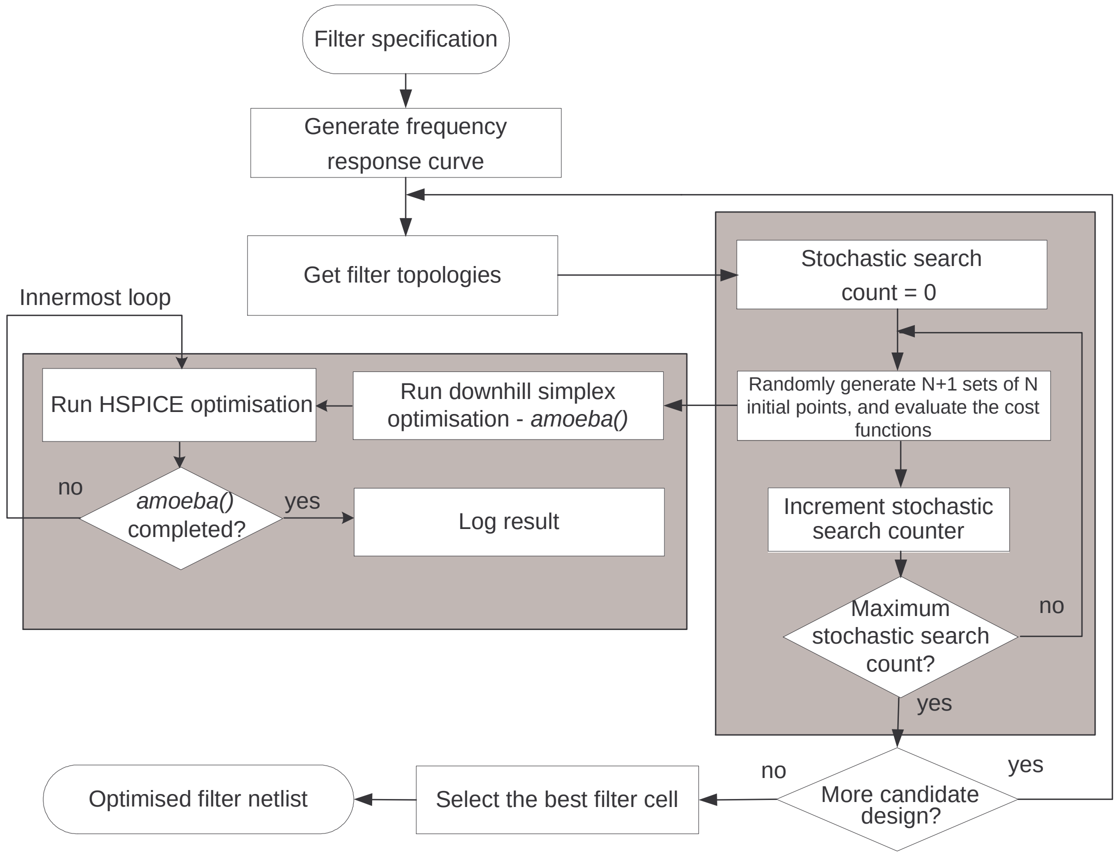
Filter specification
Generate frequency response curve
Get filter topologies
Stochastic search count = 0
Innermost loop
Run HSPICE optimisation
Run downhill simplex optimisation - amoeba()
Randomly generate N+1 sets of N initial points, and evaluate the cost functions
no
amoeba() completed?
yes
Log result
Increment stochastic search counter
Maximum stochastic search count?
no
yes
no
yes
Optimised filter netlist
Select the best filter cell
More candidate design?

Figure 3. Three-tier parametric optimization strategy.

```
entity filter is
      port (quantity Vin: real;
            quantity Vout: out real);
end entity filter;

architecture transfer of filter is
  constant a:real:=4.1589e-10;
  constant b:real:=8.6483e-20;
  constant c:real:=1.0535e-29;
  constant d:real:=6.4162e-40;
  constant num: real_vector:= (1);
  constant den: real_vector:= (1,a,b,c,d);
begin
      Vout == Vin'LTF(num,den);
end architecture;
```

Figure 4. VHDL-AMS code of a fourth-order lowpass Butterworth filter at 1GHz.

The filter topologies are implemented in a 0.35u CMOS technology, suitable for mixed-signal integrated circuit design. Figure 5a shows the general block diagram of two second-order lowpass filter block cascades selected by the synthesizer for parametric optimization. The OTA cells are labeled ‘gm1’ and ‘gm2’. Apart from the cascade, a multiple loop feedback (all-pole) structures [18] were tried in the IFLF and LF configurations shown respectively in Figure 5 b and 5c. Thus, in this example, three different types of OTA-C filter architectures were considered: folded cascade.

The OTA itself has been implemented using three different configurations [11-13]: (1) wide swing, (2) folded cascode and (3) wide-swing folded cascode. Each OTA configuration was tried with each filter architecture resulting in nine different circuits that attempt to implement the required low pass filter. All transistor widths, the bias currents as well as capacitor values were optimized using the method described in the previous section.

Table 2 shows optimization results for all the nine designs. The cost function CF was computed as defined in equation (3), where err_{total} is obtained by calculating the difference between the HSPICE’s AC response with 6 points specified as in-line data in the netlist. The 6 points are located in the critical frequency region between 0.13 GHz to 4.3 GHz. From Table 2 it can be observed that the best topology with the lowest CF is LF2, which is the leapfrog topology implemented with the folded-cascode OTA. The most accurate topology is IFLF2, i.e. the inverse follow-the-leader feedback, also implemented with the folded-cascode OTA. The topology that consumes least power is Cascade 1, i.e. the cascade of biquads using a wide-swing OTA. This topology, however, is the least accurate amongst all nine. From this observation, it can be seen that a trade-off exists between power and accuracy. This is because the accuracy is largely dependent on the circuit gain – poor accuracy is mostly due to the fact that the circuit is not producing enough gain. To increase the gain, more current needs to be drawn. This in turn, increases the power consumption. Figure 6 shows the AC responses for the nine filter topologies produced by the synthesizer.

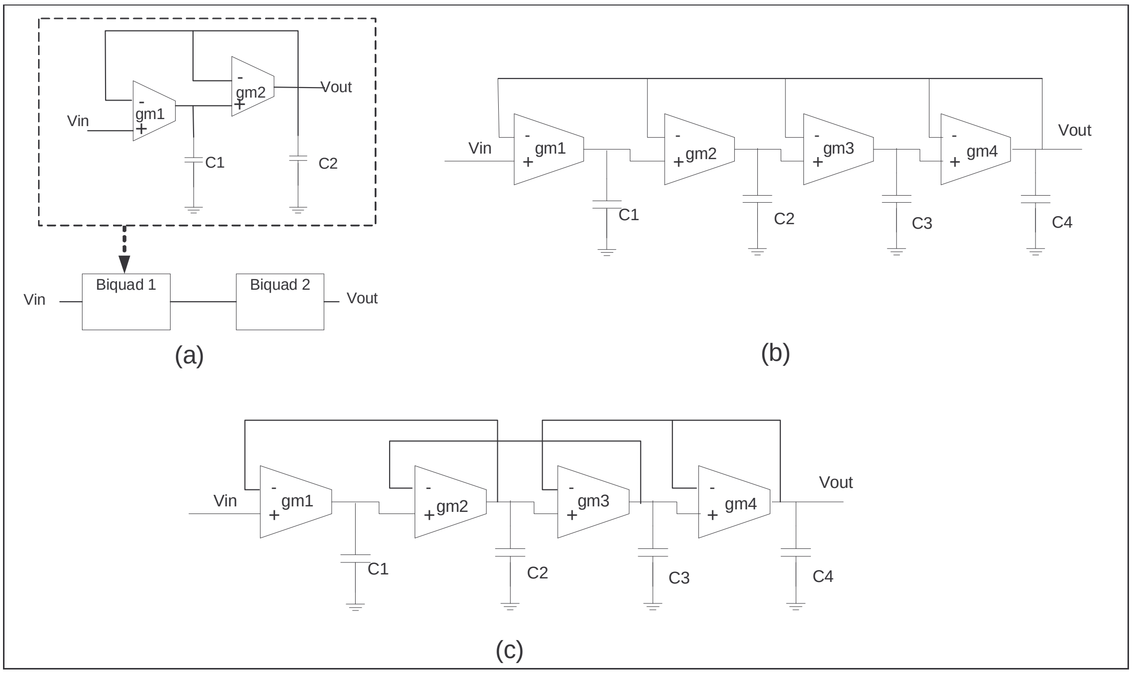

Figure 5. Implementation of a fourth-order low pass filter. (a) Using two cascaded sections of low pass biquads. All-pole multiple loop feedback structure: (b) IFLF (c) LF.

Name	No. of restarts	Evaluation no. (avrg time)	Time taken (s)	Cost function	Error function (err_{total})	Power (mW)
Cascade 1	10	194 (60.79s)	11794	0.52	0.26	3.5
Cascade 2	10	46 (46.41s)	2135	0.12	0.10	14.67
Cascade 3	10	515 (92.89s)	47839	0.25	0.18	64.3
LF 1	10	881 (65.89s)	58051	0.10	0.07	28.0
LF 2	10	1273 (52.38s)	66682	0.03	0.005	21.5
LF 3	9	1311 (225.45s)	295564	0.09	0.02	73.5
IFLF 1	8	1939 (142.38s)	276067	0.09	0.03	52.4
IFLF 2	9	1814 (163.20s)	296048	0.05	0.003	48.7
IFLF 3	4	1227 (240.84s)	295505	0.27	0.17	98.3

Table 2. Optimization results for the nine different topologies for a fourth-order lowpass filter. The number following each topology name indicates the OTA cell used in the structure, 1: Wide-swing OTA, 2: Folded-cascode OTA, 3: Wide-swing folded-cascode OTA)

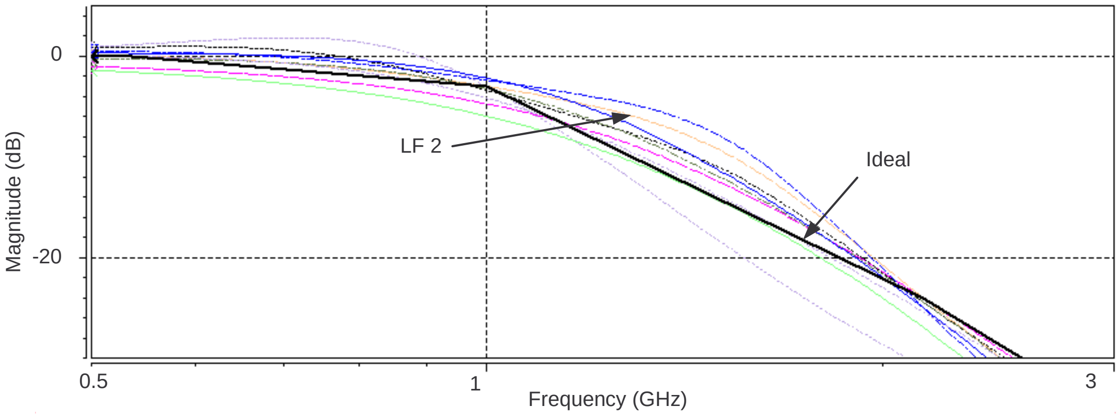

Figure 6. AC responses for the 1GHz lowpass Butterworth filter topologies listed in Table 2.

5. CONCLUSION

This contribution presents an efficient methodology for synthesis of integrated high-frequency analog filters from behavioral VHDL-AMS specifications. The synthesis algorithm used by FIST that is based on the parse tree of the behavioral VHDL-AMS has been explained. Also inclusive and vital in FIST's synthesis methodology is the architectural and parametric optimization step that involves a three-tier system that works on a selection of filter cells from a library to give the best filter circuit in terms of accuracy. As the main application for FIST is targeted for integrated high-frequency use, the filter cells as well as the optimization formulation were geared for this. The feasibility of the method has been demonstrated by the automated design of a 4th order 1GHz low-pass filter.

REFERENCES

[1] N. Dhanwada, A. Nunez, and R. Vemuri, *Hierarchical constraint transformation using directed interval search for analog system synthesis*, in *Proc. of Design, Automation and Test in Europe Conference*, pp. 328–335, 1999.

[2] A. Doboli, A. Nunez-Aldana, N. Dhanwada, S. Ganesan, and R. Vemuri, *Behavioral synthesis of analog systems using two-layered design space exploration*, in *Proc. Design Automation Conference*, pp. 951–957, 1999.

[3] A. Doboli and R. Vemuri, *Exploration-based high-level synthesis of linear analog systems operating at low/medium frequencie*s, IEEE Trans. on CAD, vol. 22, pp. 1556–1568, November 2003.

[4] A. Konczykowska, M. Bon, F. Fernandez, A. Rodriguez-Vazquez, J. Huertas, and G. Gielen, *Structural synthesis and optimization of analog circuits*, *IEEE Press*, 1998.

[5] H. Fino, J. Franca, and A. S. Garcao, *Symbolic signal flow graph methods in switched-capacitor design*, in *Symbolic Analysis Techniques*, eds: F. Fernandez, A. Rodriguez-Vazquez, J. Huertas, G. Gielen, IEEE Press, 1998.

[6] R. Ruiz-Amaya, J. de la Rosa, F. Medeiro, F. Fernandez, R. del Rio, B. Perez-Verdu, and A. Rodriguez-Vazquez, *Matlab/simulink based high-level synthesis of discrete-time and continuous-time modulators*, in Proc *DATE*, pp. 150–156, 2004.

[7] Lopez, J.A., Asensi, GD, Ruiz, R. and Kazmierski, TJ . *Automated High Level Synthesis of Hardware Building Blocks Present in art-based neural network, from VHDL-AMS descriptions*. In Proc. *ISCAS 2002*, Phoenix, 2002.

[8] Degrauwe, M.G.R., et al., *IDAC*: *An Interactive Design Tool for Analog Circuits*. IEEE Journal of Solid-State Circuits, 1987. 22(6): p. 1106-1115.

[9] Harjani, R., R.Rutenbar, and L. Carley, *OASYS: a framework for analog circuit synthesis*, IEEE Trans on CAD, 1989. 8(12): p. 1247-1266.

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

[11] Krasnicki, M., et al. *MAELSTROM: Efficient Simulation-based synthesis for custom analog cells*, in ACM/IEEE 36th Design Automation Conference. 1999.

[12] Phelps, R., et al., *Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search*, IEEE Trans on CAD, 2000. 19(6): p. 703 -717.

[13] Plas, G.V.d., *AMGIE-A synthesis environment for CMOS analog integrated circuits*. IEEE Trans on CAD, 2001. 20(9): p. 1037 -1058.

[14] Hamid, F. and T.J. Kazmierski, *FIST - a VHDL-AMS based architectural synthesis strategy for integrated high-frequency analogue filters*, in Proc. FDL. 2003. Frankfurt, Sept 2003.

[15] Nelder, A. and R. Mead, *A Simplex Method for Function Minimization*, Computer Journal, 1965. **7**: p. 308-313.

[16] *Star-HSPICE Manual*, 2000, Avant!

[17] Marquardt, D., *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, SIAM J. Appl. Math, v. 11, 431-441, 1963.

[18]Sun, Y., ed., *Design of High-frequency integrated Analogue Filters*, IEE Press (UK), 2002.