A Flexible Topology Selection Program as Part of an Analog Synthesis System

P. Veselinovic, D. Leenaerts, W. van Bokhoven

Eindhoven University of Technology
Dept. of Electrical Engineering, EEB
P.O.Box 513, 5600 MB Eindhoven
The Netherlands

F. Leyn, F. Proesmans, G. Gielen, W. Sansen

Katholieke Universiteit Leuven
Dept. of Electrical Engineering, ESAT-MICAS
Kardinaal Mercierlaan 94, B-3001 Heverlee
Belgium

Abstract

The task of a topology selector within an analog synthesis system is to find the best available analog circuit topology out of a library for a given set of input specifications. The proposed selection method consists of a combination of two approaches: procedural filtering and rule-based filtering. The procedural filtering consists of two consecutive phases based on boundary checking and interval analysis. Such a combination of different sorts of filtering is a new technique that allows an optimal trade-off between selection accuracy and required selection time. The tool that implements the method is technology independent and fully open towards newly added design knowledge.

1 Introduction

For the synthesis of analog integrated circuits, an hierarchical design strategy is nowadays being used in most programs [1,2]. The design of higher-complexity modules (e.g. A/D converters) is translated into the design of smaller lower-complexity circuits (e.g. comparator) and ultimately devices. In between each of the hierarchical levels the design process consists of several steps such as topology selection, circuit sizing, layout generation, extraction and verification. Topology selection has been recognized as the first of those consecutive steps.

The goal of topology selection is to search through the set of candidate topologies in a library that implements the required circuit behavior and to find the topology that best matches the input performance specifications in the specified technology process. The set of input specifications can be provided either by a human designer who uses such a tool, or can be derived from a synthesis step at a higher hierarchical level in an automated analog synthesis system. The process can be repeated hierarchically in the sense that a topology is built up from lower -level blocks, for each of which later on in the design process again a lower-level topology has to be selected, and so forth.

Although topology selection is an inherent part of analog synthesis, many programs published in the literature so far do not cover this problem. Programs that do handle it, such as OPASYN [3], OASYS [4], or the stand-alone tool HECTOR [5], use heuristic rules to decide between the different predefined alternatives. The selection between different related topologies in [6] has been integrated within the sizing process, by adding binary variables that control the topology configuration as additional optimization variables. However, none of these techniques makes explicit use of quantitative data about the obtainable performance ranges of the different candidate topologies to carry out the selection. The use of such information is, however, essential for the selection of a good topology, and the selection of a good topology is as important to obtain a high-quality design solution (for the lowest power and area consumption) as the use of (global) optimization when sizing the selected topology afterwards. Our aim is to use quantitative performance space data to select the most promising topology candidate, which will then be sized by a separate sizing tool. The use of performance-space data also reduces the need for CPU-time-expensive redesign iterations - decreases the likelihood that a selected topology later on in the design process turns out not to be able to meet the input specifications.

This paper introduces a topology selection composed of analytical filtering (based on boundary checking and interval analysis) and rule-based filtering to obtain a flexible program that combines selection precision with short selection time. In section 2, the topology selection program is situated as essential part of an overall analog synthesis system. Section 3 describes the method used for the selection and covers three filter parts. A detailed design example and implementation aspects are discussed in section 4. Conclusions will be drawn in section 5.

2 Topology selection within analog synthesis

The topology selection program presented in this paper is an essential part of the analog module synthesis system (Fig.1), that is presently under development [7]. The main tools in the system are the following programs: topology selection, sizing and optimization, verification, and analog layout generation. The key requirements for the system are modularity, openness and flexibility, which means that the tools are independent and exchangeable and interact through a common database under control of a design controller (which manages the design flow), that new design knowledge and new topologies as well as new technology processes can be added easily, and that different design styles are supported.

To accomplish this, two libraries are used in the system, namely a technology library and a cell library. Technology independence (e.g. 2.4 cmos and 0.7 cmos) is established by storing files with the actual values for technological process parameters in the technology library. The cell library contains descriptions of all module and circuit topologies that are known to the system and contains all information that is needed to carry out the different synthesis steps. In order to be able to easily add new cells, the topology selection method must be

This research was part of a project with ESA-ESTEC (No9890/92/NL/GS)

independent of the current and the future contents of this cell library. At the same time, the topology selection program can only select among topologies that are already stored in this cell library and the information needed to obtain the performance range at run time must be stored with each topology in the cell library.

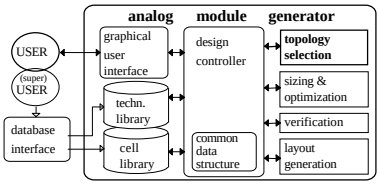

Fig. 1. Logical model of an analog module synthesis system.

The topology selection program will reject all cells from the library that do not have the required behavior or are not able to meet the input specifications. Depending on the tightness of these specifications the outcome will be that none, one or more topologies are considered as valid candidates. In the latter case, the selection program must also rank all remaining candidate topologies and choose the best one among them.

Redesign has to take place whenever the verification tool detects that the circuit under design fails to meet the input specifications. Depending on the fault recovery strategy, it might then be necessary to redo topology selection, since the previously selected topology might not have been a good one after all. This might happen when the required circuit specifications fall on the edge of what a particular topology can offer. Topology selection during this redesign phase differs from the regular design selection because of the additional information about the reasons of the failure that the selection program can use in order to come up with a better solution. The sizing of a selected topology is carried out by a separate sizing program (Fig.1).

3 Method of topology selection

3.1 Overview of the applied method

In order to avoid design iterations in the synthesis system that include several design phases (sizing and optimization or even layout generation), the method for topology selection should be reliable and accurate. Furthermore, the selection method should be fast enough to allow interactive use. These two constraints are usually in conflict and our aim was to explore possible trade-offs and to find a method that offers the designer full flexibility in finding a good compromise.

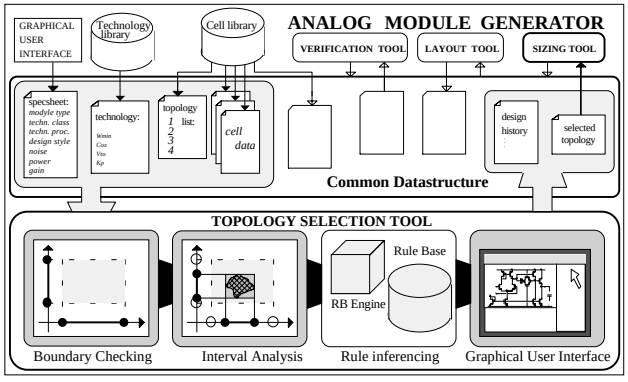

Fig. 2. Diagram of the topology selection program.

The filtering sequence (Fig.2) is a combination of procedural and knowledge-based approaches. These filters are a boundary checking (BC) filter, an interval analysis (IA) filter and a rule-based (RB) filter, respectively, and will be discussed in detail in the following subsections. The modular concept of the selection filters provides two major advantages. Firstly, it allows easy extension of the cell library. Secondly, it allows tuning of its performance by means of accuracy-speed trade-offs: The two procedural filters (BC and IA) differ in their complexity, accuracy of their outcomes and their speed. On the other hand, the procedural filters and the knowledge based filter differ in their approaches and may be complementary.

The sequence of first applying BC, followed by IA and then RB filtering ensures efficiency of the selection process in terms of the overall required CPU time and the obtained accuracy. The fast rejection of non-feasible topology candidates at the beginning will reduce the set of topology candidates to a number acceptable from the perspective of the CPU time required to perform IA. The rule-based ranking and in case of redesign reranking, provide us with a reasonably good final selection.

3.2 Selection filter based on boundary checking

In analog IC design for each circuit (e.g. an opamp) there is a finite, although relatively large set of possible topologies (e.g. different opamp schematics). Each circuit topology can be characterized by an analytic model represented by a set of design constraints. The design constraints are, in general, nonlinear equations and inequalities, giving the relationship between design variables, i.e. circuit resources such as transistor sizes W and L, and the performance behavior of the topology, i.e. circuit properties (e.g. gain, bandwidth, power consumption, etc.). For a circuit topology and a specified behavior, the circuit resources can only take their values from within certain intervals. In the case of transistor sizes, the lower boundary will be dictated by the chosen technology process while the upper boundary will be restricted by requirements of chip area minimization and layout aspects, resulting in for instance an allowed transistor width between 0.8 and 1000 μm. As a result, the different circuit performance characteristics for the same topology will also show values within some intervals. These feasible performance intervals of course depend on the selected technology process.

For a given design application, the user imposes requirements on the performances that the circuit must achieve. These are the input specifications that must be met by the topology that is to be selected. For the example of an opamp, there are more than 20 characteristics for which the designer can provide a specification, but depending on the application only a subset of these will really be specified. (This holds, especially, when specifications are translated automatically from an upper hierarchical level by an automated design system.) The specifications are intervals, corresponding to the range of values for each particular performance characteristic that the designer can accept for his application. This specification interval can be finite or open, and a special case of a finite interval is a single value (e.g. 'Phase margin has to be at least 60°'; 'Power consumption has to be less than 1mW'; 'Output range has to be -3V to +3V';' Gain has to be exactly 100').

Thus both the feasible performance intervals and the required performance specifications can be represented by intervals; different circuit topologies can be evaluated by comparing their feasible performance intervals; topology selection can be performed by comparing the specifications with the topologies' feasible performance intervals. The calculation of these inter-

vals, which indicate the minimal and maximal possible values achievable by that topology, can be done at run-time or on forehand. In the latter case the precalculated intervals can be stored in the cell library together with the circuit topology, which reduces the required CPU time at run time but has the drawback of being technology dependent. The actual selection process is then carried out as follows. Each topology for which there is at least one performance characteristic that shows no overlap between the specification interval and the corresponding feasible interval covered by that topology must be rejected, as this topology is definitely not able to meet that particular specification in the given process. In this way, already a large number of inappropriate topologies can be eliminated. This technique is called filtering based on boundary checking.

Fig.3a shows a 1D comparison of four topologies (T_1 till T_4) based on the feasible interval for one performance parameter *i* with a specification range P_i. The selection process rejects T_2 whose feasible interval does not overlap with P_i, and ranks the remaining topologies as T_3, T_1, T_4 depending on the size of the overlap region. Fig.3b depicts another comparison of four topologies, 2D - this time based on feasible intervals for two performance parameters P_1 and P_2. T_1 and T_4 are rejected and the ranking of the remaining topologies is T_3, T_2.

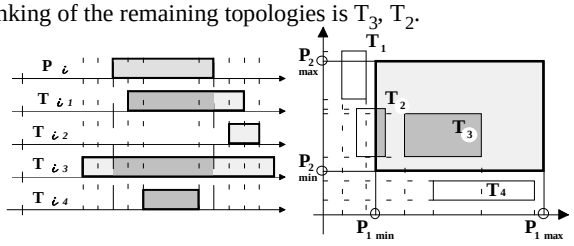

Fig. 3. Boundary checking illustrated for one and two performance characteristics, respectively.

In a real design example, as said before, the topology will have many more performance parameters, say n, and the feasible intervals for these performances will form an n-dimensional bounding box. The BC filter then checks whether there is a cross section with the box of the specifications or not. Since the filter essentially considers each performance parameter independently of all the others, BC is simple and fast. Topologies that pass BC do not necessarily meet the specifications after sizing since the filter does not take the correlations between the different performances into account (an opamp can not obtain the maximum gain at the same time as the maximum bandwidth). In other words, the real feasibility space is a subspace of the bounding box used in the BC filter. Since the relation between performance parameters is contained within the circuit's design equations, in order to achieve a more accurate definition of the feasibility space we have to take these design equations into account, which brings us to the concept of IA.

3.3 Selection filter based on interval analysis

To allow a more accurate selection between the remaining candidate topologies, the feasible intervals for all performances, taking into account all their mutual influences or correlations, have to be calculated for each of these topologies. All design (in)equalities for the performance variables bounded by their specification values have to be solved simultaneously. Then the obtained intervals will depict the complete solution space and again unfeasible topologies can be eliminated. This technique is called filtering based on interval analysis.

The key of the technique is to calculate the solution space. A method has been reported in [8] to find the whole solution space for an underconstrained system of linear inhomogeneous (in)equalities, together with its application to analog circuit design. Although the design equations are usually nonlinear, nonlinear functions can be approximated by a piecewise linear (PL) format and then all solutions can be found. In either case the solution space is described by a linear combination of a set of vectors (corner points of the solution space). This technique is also very useful in hierarchical design [9]. If the solution space is empty at the higher design level, it means that there is no need to explore the lower levels (e.g. subcircuits) since no feasible design is possible.

The improvement that IA brings over BC is depicted in the examples of Fig.4 and 5. Fig.4 differs from Fig.3 in that the correlation (functions f_1 and f_2) between the performance parameters P_1 and P_2 has been added. This correlation between parameters is expressed by the model equations.

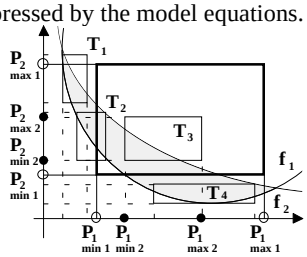

Fig. 4. Feasibility check with the relations between two parameters.

For the topology T_3, Fig.5 shows user-specified intervals (1) and the initial intervals (2) for two performance parameters as stored in the cell library, together with intervals (3) for those parameters after an evaluation took place taking into account the relationship between P_1 and P_2. Point d belongs to the solution space of the T_3 topology, while point c is outside of it, in contrast with simple BC that would accept both c and d.

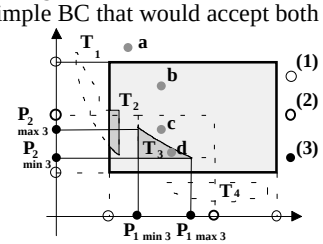

Fig. 5. Combined results of boundary checking and interval analysis.

The accuracy of the obtained solution space depends on the accuracy of the models (equations) used to compute the solution space. Since the calculation method itself (but not its performance) is independent from the model used to describe a topology, it is possible to achieve acceptable selection accuracy with moderate CPU-time and memory usage by using higher-level description models for analog modules and circuits. Higher-level IA avoids possible redundancy (with sizing) introduced by considering the solution space at the lowest level. Eventually, however, we have to deal with a topology at the device level, and then we have to manage models with a relatively large number of parameters, which drastically increases the CPU-time and memory consumption. By reducing the number of parameters taken into account for IA, we can reduce the required CPU time at the expense of selection accuracy. Here a compromise takes place: a number of parameters is modeled for IA and a number of parameters is evaluated by the rule base.

It is well known that certain topologies, regarding a specific performance, have advantages over others. A typical example is the use of cascoded versus noncascoded subblocks depending

on the gain requirements. Using if-then-else rules to decide when to apply one or the other is usually much more efficient than to explore the complete solution space in order to find the most appropriate topology. It is therefore appropriate to include such general design knowledge as rules into the rule base of the knowledge-based filter.

3.4 Rule-based selection filter

Up till now, starting from all topologies in the cell library that implement the required circuit behavior, the first filter based on BC has rejected all obviously unfeasible topologies. The second filter based on IA has found the solution space for the remaining topologies and then decided whether or not to reject them. Eventually there might be several candidate topologies left. The final step is to rank these candidates using the information about the topology's property attributes. This ranking has been implemented as a knowledge-based filter based on heuristic selection rules [3-5]. Indeed, a knowledge-based approach is suitable for conflict resolution problems. Since the ranking of the remaining topologies that satisfy the specifications with their solution space can be considered as a conflict resolution, this ranking can be performed efficiently by means of rules.

A strong reason to use the knowledge-based filter are the redesign iterations in the analog design system of Fig.1. When a new topology has to be selected for the same specifications, the procedural filters do not have to be rerun since they will return exactly the same result. The rule-based filter on the other hand can use the additional failure information about the specifications that were not met by the original topology that caused the redesign, to rerank the remaining topologies and select a new best topology. A redesign rule can then for instance state that if the gain requirement was not met for a noncascoded topology, then the cascoded version has to be tried next.

4 Example of topology selection

4.1 Implementation aspects

The topology selection program has been implemented in the C language for the procedural filters and in Prolog for the rule-based filter, on a SUN workstation and it is presently being integrated into the analog synthesis system of Fig.1. The program is built up completely in a modular way (see also Fig.2) and all three filters take different data files from the cell library as inputs, besides the overall inputs like user specifications, technology data, etc. The inputs are ASCII text files. The BC data file contains the boundary values for the feasible performance intervals, which are calculated in advance and stored in the cell library for well established designs (fixed cells) or which can be calculated at run time (parameterized and custom cells). The IA data file contains all kinds of data needed for interval analysis, related to a specific topology. Openness towards new design is fully respected: new topologies for a known function block can be added to the cell library just by adding their necessary data files. No recompilation or code update of the selection program are needed. The same holds for adding new technology processes (e.g. 0.xx μm) within one of the covered technology classes (e.g. CMOS).

The model development process can substantially be facilitated by the use of tools such as ISAAC [10] and DONALD [11]. The IA model file contains information which is at the runtime transformed in the following form:

$$a_{11} \cdot x_1 + a_{12} \cdot x_2 + \ldots + a_{1n} \cdot x_n + b_1 \quad \rho_1 \; 0; \quad \rho_i \in \{ =, \leq \}, \; i = 1,\ldots,m$$
$$\vdots$$
$$a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \ldots + a_{mn} \cdot x_n + b_m \quad \rho_m \; 0;$$

The coefficients a_{ij} are constants whose values depend on the input specifications, the particular topology and the specified technology process. The unknowns x_j are the design parameters that can take values from intervals, the boundaries of which are defined by input specifications, properties of the particular topology and the technology parameters used. The solution obtained by IA is in the form of a linear combination of vectors, depicting the solution space for the unknowns x_j. Due to the nature of the algorithm, it is advisable to keep the number of unknowns relatively small in order to limit the time required to solve the system. One of the severe constraints for the model is its required linearity. As said above, however, nonlinear equations can also be handled by approximating them with piecewise linear models [8].

4.2 Practical design example

The particle-detector front-end circuitry of Fig.7 consisting of a CSA (charge sensitive amplifier) and a PSA (pulse shaping amplifier) is chosen as example here. The CSA and PSA are from the classes of modules (amplifiers and filters) that the analog module synthesis system will be able to generate. The CSA is built of an OTA (operational transconductance amplifier), while the PSA is built of several OPAMPs, in an hierarchical way.

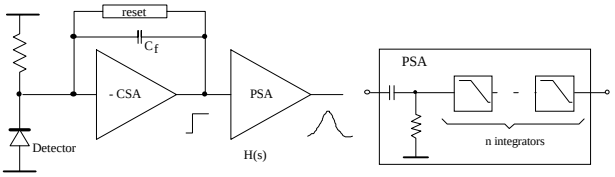

Fig. 7. Particle-detector front-end circuitry as an example.

The performance of the read-out module is controlled by 5 major design parameters: the width, length and bias current of the input transistor, the shaping time and the number of integrators in the shaping amplifier. The key input specifications are noise performance, power consumption and speed for a specified detector capacitance. All of these specifications are directly affected by the choice of the values for the design parameters. The input transistor's width and length have practical limits imposed either by technology or by the performance of other parameters, the bias current is limited by the power consumption requirement, the shaping time is an input specification and limited as well, and finally the number of integrators is a property of the particular topology. To keep the model sufficiently accurate, yet simple, ten input parameters (Fig.9) were chosen to be included in the interval-analysis model. This model has been chosen taking into account design knowledge and the relative importance of some parameters over the others. Conceptually, the model looks as follows:

A : specs, topology's design parameters, technology parameters
(speed) BandWidth= f_1(A);
(noise) EquivalentNoiseCharge= f_2(A);
(power) DissipatedPower = f_3(A);
(design equations): f_i(A)=0 i = 4,...,m.

In this model, the design parameters vary from topology to topology in two different ways. Firstly, the parameters are different for different topologies (e.g. different expressions for the

speed). Secondly, the boundary values for the intervals for the parameters differ from one topology to the other. The total number of model variables in the used models was >25.

The topology selection program has then been tested for a cell library containing 8 cells with different design styles for CSA-PSA topologies. There were 4 topologies with PMOS and 4 with NMOS input transistors, called P_i and N_i, respectively (i = 1,...,4). The number of topologies that pass the BC or IA filter depends on how relaxed or tight the input specifications are. The time required for the IA filter is proportional to the number of topologies remaining after the BC filter. The ranking returned by the IA filter differs from the initial ranking returned by the BC filter, because of the shrinking of the intervals during the IA filtering. Relaxed specifications are used here to visualize the results of IA. Fig.8 shows the lower noise boundary for a given power consumption and vice versa for the eight different CSA-PSA topologies. Those are the corner points of the solution space for the given requirements, obtained by IA. The relevant rules for ranking in the rule-based filter consider general knowledge about noise performance and required chip area.

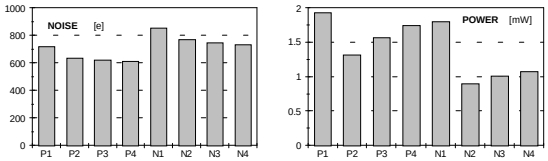

Fig. 8. Some corner points of the solution space determined by interval analysis for the different CSA-PSA topologies

Two sets of input specifications together with the obtained results are summarized in Fig.9. These two sets represent optimized noise and power consumption cases, respectively. In both cases the CPU time was below one second, most of it being spent during the IA filtering. The example shows the BC has rejected 2 and 3 cells due to their non-feasibility, for the two cases respectively. Furthermore, IA finds that 4 and 2 of the remaining cells, respectively, can not fulfill all the requirements simultaneously. The difference in ranking after IA and RB is due to the nature of the evaluation: IA evaluates the size of allowable intervals, and RB evaluates cell attributes. The procedural filters calculate the ranking value for a topology j according to:

$$rv_j = \sum_{i=1}^{N} \left(\frac{2 * \left\| parameter.\max_i \right| - \left| parameter.\min_i \right\|}{\left\| parameter.\max_i \right| + \left| parameter.\min_i \right\|} \right)$$

input specs	case 1	case 2	units
Cd	= 80E-12	= 60E-12	F
I_leak	= 10E-9	= 5E-9	A
mxm_peak_t	< 1.5E-6	< 1.5E-6	s
count_rate	> 200E3	> 200E3	Hz
noise	< 780	< 800	electron
pos_sppl	= 3	= 3	V
neg_sppl	= -3	= -3	V
gain	= 40	= 40	mV/MeV
out_range	> 1.4	> 1.4	V
power_dis	< 2E-3	< 1E-3	W

Nr and rank	case 1	case 2
BC	P4,P3,P2,P1,N4,N3	N2,P3,P4,P1,P2
IA	P3,P2	N2,P2,P1
RB	P2, P3	P2,N2,P1

Fig. 9. Results of each filter's performance for 2 sets of input specs.

The final rank is obtained by rules addressing the topology's properties such as complexity, and making a tradeoff between noise and power performance. One example rule used for final ranking within the RB has the following form:

```
IF { nr_integrator.cell_A < nr_integrator.cell_B } THEN
cell_A before cell_B
```

Another set of experiments were conducted with input specifications from a very wide range. The outcome shows that: the selected topology, the number of not rejected topologies and their rank vary significantly with the variation of the input specifications. Furthermore, it has been shown that each of the 8 used topologies has a region within the solution space where it is dominant or superior to the others. The relatively large number of input parameters, as well as the number of dimensions of the solution space makes it difficult to visualize such a result. Easy explainable results, however, are yet available in numerical form.

5 Conclusions and future work

A new method has been presented for topology selection within an automated analog synthesis system. The method consists of the subsequent application of two procedural filters, based on boundary checking and interval analysis respectively, followed by a rule-based filtering. This combination allows a trade-off between selection accuracy and required selection time. Practical evaluation results of this method have been presented and show an acceptable performance, which is presently being improved further on towards better speed and/or accuracy. The topology selector is currently being integrated within a complete automated analog synthesis system.

References

[1] G. Gielen, K. Swings, W. Sansen, "Open analog synthesis system based on declarative models," chapter 18 of "Analog circuit design" (edited by H. Huijsing, R. van der Plassche, W. Sansen), Kluwer Academic Publishers, pp. 421-445, 1993.

[2] E. Malavasi et al., "A top-down, constraint-driven design methodology for analog integrated circuits," chapter 13 of "Analog circuit design" (edited by H. Huijsing, R. van der Plassche, W. Sansen), Kluwer Academic Publishers, pp. 285-324, 1993.

[3] H. Koh, C. Séquin, P. Gray, "OPASYN: a compiler for CMOS operational amplifiers," *IEEE Trans. on CAD*, Vol.9, No.2, pp. 113-125, 1990.

[4] R. Harjani, R. Rutenbar, L. Carley, "OASYS: a framework for analog circuit synthesis," *IEEE Trans. on CAD*, Vol.8, No.12, pp. 1247-1265, 1989.

[5] K. Swings, S. Donnay, W. Sansen, "HECTOR: a hierarchical topology-construction program for analog circuits based on a declarative approach to circuit modeling," *procs. CICC*, pp. 5.3.1-5.3.4, 1991.

[6] P. Maulik, L. Carley, R. Rutenbar, "A mixed-integer nonlinear programming approach to analog circuit synthesis," *procs. DAC*, pp. 693-703, 1992.

[7] H. De Donker, S. Donnay, K. Lampaert, F. Proesmans, K. Swings, G. Van der Plas, G. Gielen, W. Sansen, D. Leenaerts, P. Veselinovic, S. Buytaert, M. Buckens, K. Marent, C. Das, "Analog module generator for space applications", *procs. ProRISC*, pp.141-146, 1993.

[8] D.M.W. Leenaerts, J.A. Hegt, "Finding all Solutions of Piecewise Linear Functions and Application to Circuit Design", *Int'l Journal of Circuit Theory and Applications*, Vol 19, pp.107-123, 1991

[9] D. Leenaerts, "TOPICS: a new hierarchical design tool using an expert system and interval analysis", *procs. ESSCIRC*, pp.37-40, 1991.

[10] G. Gielen, H. Walscharts, W. Sansen, "ISAAC: a symbolic simulator for analog integrated circuits," *IEEE Journal of Solid-State Circuits*, Vol.24, No.6, pp. 1587-1597, 1989.

[11] K. Swings, G. Gielen, W. Sansen, "An intelligent analog IC design system based on manipulation of design equations," *procs. CICC*, pp. 8.6.1-8.6.4, 1990.