Parameter Domain Pruning for Improving Convergence of Synthesis Algorithms

Hua Tang, Alex Doboli
Department of Electrical and Computer Engineering,
State University of New York at Stony Brook, Stony Brook, NY, 11794-2350
E-mail: htang, adoboli@ece.sunysb.edu

Abstract— **This paper presents a parameter domain pruning method. Parameter domain pruning aims to identify parameter sub-domains that are more likely to produce feasible and good design solutions. These parameter sub-domains are found using the proposed Simplified Affine Transforms (SAT) operators. Selected variable sub-domains can then be used as input to exploration-based synthesis tools and help improve the convergence of synthesis algorithm.**

I. Introduction

Analog IP cores are essential building blocks for modern systems-on-chip (SOC). Designing analog circuits is a difficult and cumbersome task that requires extensive designer expertise. In fact, many reports [1] suggest that designing analog circuits is the bottleneck of SOC design, in spite of the much lesser size of the analog part as compared to the digital modules. Analog synthesis tools are necessary to boost design productivity, but also to enlarge the group of designers that can tackle analog circuits. Hence, analog synthesis tools should be fully automated, and require minuscule amounts of designer knowledge as inputs.

Present analog synthesis methods fit into two categories: solving based approaches and exploration based techniques. In solving based methods, the synthesis problem (including design requirements and objectives) is expressed as a mathematical model, such as posynomials [2], and then solved using a mathematical solver. For instance, Hershenson *et al* [2] present a geometric programming based technique for synthesis of OpAmp. Solving methods provide optimal design parameter values, but they require a cumbersome, manual process to build the model of a system. Extensive analog knowledge is needed to create models. Also, design parameters are limited to small ranges to ensure an optimal solution.

In contrast, exploration based synthesis involves less designer effort, and is more flexible in tackling various types of applications. A variety of traditional optimization techniques are used, like simulated annealing [3] [4], genetic algorithms [5] and tabu search [6]. Exploration based synthesis needs sampling of a large number of solution points, thus it might be quite slow and experience convergence difficulties. A solution to improve convergence is to use design knowledge like limiting parameter values to small ranges.

As can be seen, both categories of analog synthesis tools needs to have limited design parameter ranges as inputs as they are critical for improving synthesis convergence because they narrow down the solution space. From exploration point-of-view, the requirement is equivalent to knowing a priori a good solution space region, which is likely to contain feasible solutions and good solutions. However, in general, it might be difficult to know these parameter ranges, especially for new analog circuits and systems. A problem is that these parameters ranges are not obtained automatically but rather from accumulated designer's experience.

In this paper, we present a parameter domain pruning method to automatically identify parameter sub-domains that are more likely to produce feasible and good solution points. The method splits variable domains into intervals, and estimates the ranges for a certain performance metric, which has an accurate closed form expression of the design parameters. Intervals which lead to wide deviation from the desired performance ranges are discarded, as they would require more exploration effort than intervals with a more compact performance range. Performance ranges are calculated using the proposed Simplified Affine Transforms (SAT) operators. Pruned parameter domains are then given as inputs to the synthesis algorithm and, as experiments show, they greatly improve the convergence of synthesis tools.

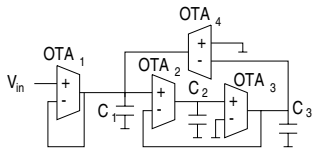

Fig. 1. **Third order elliptic lowpass filter**

The paper is organized as six sections. Section II presents previous work and our motivation. Section III presents the considered analog synthesis flow. Section IV discusses the proposed SAT. Section V gives some experiments. Finally, we present our conclusions.

II. Previous Work

The convergence of heuristic exploration algorithms critically depends on having reduced variable domains that contain good solution points. In analog synthesis tools [2], parameter are limited in small domains and these domains are then given as inputs without giving information on how these parameter domains are obtained. In this paper, we propose an Interval Arithmetic (IA) based technique to automatically prune parameter domains.

Interval Arithmetic [14] has been used in various areas of circuit and systems [7] [8] [10] [11] [12]. Traditionally, it is mainly used for circuit tolerance analysis [7] [8]. Affine Arithmetic (AA) [15], a further extension of IA, has been used for global optimization of mathematical functions [9]. But it is limited to functions of two variables, which is due to the heavy computation overhead of AA. AA has also been used to size analog circuits [10], but current method is limited to very small circuits, with only two design components and one performance metric to be optimized. [11] used AA technique to estimate the precision effects of floating point operations in DSP applications. Recently, C. Grimm [12] proposed AA technique to estimate the output of a circuit so that circuits with unexpected output are discarded.

In the context of analog synthesis, there are usually more design variables and more performance metrics to be simultaneously optimized. Also, the performance metrics can not all be closed-form expressions of design parameters. Considering these two factors, we can not directly use IA for synthesis and optimization as in [10]. Instead, we propose to use IA-based techniques to first prune the parameter domains, and then let the exploration algorithm to do

0-7803-8834-8/05/$20.00 ©2005 IEEE.

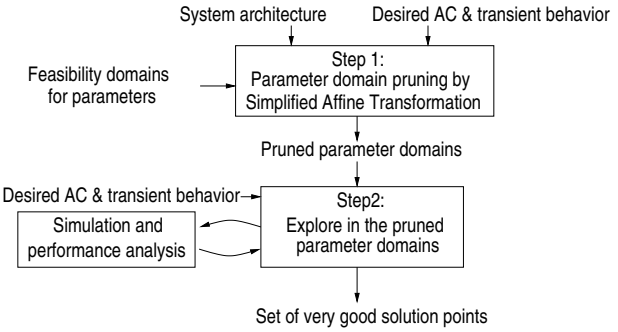

Fig. 2. **Analog synthesis methodology**

synthesis on the identified parameter sub-domains, in which there is a higher chance that good solutions exist. In the exploration phase, we can then optimize all the performance metrics, including those that do not have closed form expressions.

III. Revised Analog Synthesis Flow

Figure 2 presents the considered analog synthesis flow. We will target analog filter synthesis in this paper. Inputs to the flow are system architectures and performance requirements in AC and transient domains. Architectures are netlists of active circuits, such as OpAmp and GmC, and passive elements, like resistors and capacitors. Synthesis output is a set of performance satisfying system architectures having all their resistors and capacitors sized.

The cost function for the synthesis problem is defined as:

$$Cost : D_1 \times D_2 \times ...D_n :\rightarrow CoDomain,$$
$$Cost = f(p_1, p_2, ..., p_n) =$$
$$\sum_i F_i(p_1, p_2, ..., p_n) + \sum_j G_j(p_1, p_2, ..., p_n)$$

where functions F_i are continuous and differentiable. Functions G_j do not have a closed-form expression, but instead, are described by algorithms defined over the simulation data for the architecture. $D_i = [min_i, max_i]$ is the domain of variable p_i expressing values feasible for design. Symbols p_i are the design parameters to be searched during synthesis. For example, for the filter synthesis, one tries to minimize the magnitude error with respect to the ideal magnitude, minimize the phase nonlinearity etc. Magnitude response can be accurately expressed with the transfer function. But for phase nonlinearity, it is hard to find an accurate closed form expression. Thus, it belong to a G_j function and can be accurately calculated from simulation data.

The considered analog synthesis flow differs from traditional exploration based synthesis in that it prunes away unattractive parameter domains before searching for good quality design points. Parameter domain pruning eliminates parameter regions that largely violate performance constraints, thus determine large cost function values. These regions are unlikely to include feasible solutions or good solutions. Hence, these regions decrease synthesis convergence by "wasting" sampling steps over many unattractive points. Section IV discusses the proposed SAT operators for parameter domain pruning. After step 1, step 2 use exploration based algorithms to find global optimal solutions in the identified parameter sub-domains.

IV. Parameter Domain Pruning using Simplified Affine Transformations

Parameter domain pruning starts by considering the feasibility domain of each parameter. The feasibility domain indicates the value range to which a parameter pertains for a certain technology. For example, for third order elliptic filter shown in Figure 1, Gm values of the GmC-s circuit are in the continuous range $[1.0e-6, 1.0e-2]$ and capacitor values are in the range $[1.0e-15, 1.0e-11]$ [13]. Even though feasibility domains are large, they must be fully analyzed to find good quality design points. In the proposed method, Interval Arithmetic [14] is used for parameter domain pruning. To improve speed and precision, we developed a variant of Affine Transformation called Simplified Affine Transformation (SAT).

The pruning step considers a simplified form of the cost function $cost'$. $Cost'$ includes only some of the F_i terms without incorporating any of the G_j functions. The reason is that interval arithmetic operators are applicable only to closed-form expressions that are continuous and differentiable, and are difficult to be used for performance parameters expressed algorithmically, like function G_j. The terms in cost function $Cost'$ are found through symbolic computations.

For example, one of the function F_i is to minimize the magnitude error of the synthesized filter with respect to the ideal magnitude response. Magnitude response can be found as a closed form expression of all the design parameters. Thus, we can compute the ranges of the magnitude response for different combinations of intervals of the design parameters. In particular, parameter ranges were selected such that the co-domain of the cost function $Cost'$ was within a range centered around the ideal magnitude response: for each frequency point f_i, the cost function co-domain had to pertain to the range $Mag(TF_{ideal}(f_i)) \times (1-\epsilon, 1+\epsilon)$, where $Mag(TF_{ideal}(f_i))$ is the ideal magnitude value at frequency f_i and ϵ is a small positive value. If such domains are found then exploration becomes easier, as any parameter combination produces a magnitude response in the neighborhood of the ideal magnitude. Performance violations due to the magnitude response are thus avoided. This section presents the SAT operators based on Interval Arithmetic for computing the co-domain of $Cost'$.

IA operators are defined for any two intervals $I_1 = (a_1, b_1)$ and $I_2 = (a_2, b_2)$ as following [14]:

$$I_1 + I_2 = (a_1 + a_2, b_1 + b_2);$$
$$I_1 - I_2 = (a_1 - b_2, b_1 - a_2);$$
$$I_1 \times I_2 =$$
$$(\min\{a_1\ a_2, a_1\ b_2, b_1\ a_2, b_1\ b_2\}, \max\{a_1\ a_2, a_1\ b_2, b_1\ a_2, b_1\ b_2\});$$
$$\tfrac{I_1}{I_2} = (a_1, b_1) \times (\tfrac{1}{b_2}, \tfrac{1}{a_2}), \text{ if } 0 \notin (a_2, b_2).$$

Specific IA operations are also defined for square root, trigonometric functions, and so on [14].

The main limitation of IA operators is their over-estimation of the exact range bounds [15]. This is because of loosing dependencies between sub-expressions, as IA operators consider each operation and sub-expression as being totally independent. Accounting for range over-estimation is important, if IA were to be used for pruning unattractive variable ranges. Estimation accuracy can be improved by using Affine Transformations [15]. Considering its superior precision [7] [15], we decided to employ a variant of affine transformation for parameter domain pruning.

Definition[15]: Affine Transformations (AT) express the range $I = (a, b)$ as $I = cv + rad \times \epsilon$, where $cv = \frac{a+b}{2}$, $rad = \frac{b-a}{2}$, and $\epsilon \in (-1, 1)$. Value cv is called central value, value rad is the radius, and symbol ϵ is the noise. Each range introduces a new noise symbol. Following interval operators are defined for AT:

$$I_1 + I_2 = cv_1 + cv_2 + rad_1\ \epsilon_1 + rad_2\ \epsilon_2;$$
$$I_1 - I_2 = cv_1 - cv_2 + rad_1\ \epsilon_1 - rad_2\ \epsilon_2;$$
$$I_1 \times I_2 = cv_1\ cv_2 + rad_2\ cv_1\ \epsilon_1 + rad_1\ cv_2\ \epsilon_2 + rad_1\ rad_2\ \epsilon_3;$$
$$\tfrac{I_1}{I_2} = cv_1\ cv_2' + rad_2'\ cv_1\ \epsilon_1 + rad_1\ cv_2'\ \epsilon_2 + rad_1\ rad_2'\ \epsilon_3,$$

where cv_2' and rad_2' are the central value and radius of the interval $(\frac{1}{b_2}, \frac{1}{a_2})$, and $0 \notin (\frac{1}{b_2}, \frac{1}{a_2})$.

```
for ∀ subintervals I1 of parameter 1 do
 for ∀ subintervals I2 of parameter 2 do
...
  for ∀ subintervals Ik of parameter k do
   (min,max) = evaluate the function for the current subintervals for parameter 1,...k;
   if min < global_min then
    global_min = min;
   if max > global_max then
    global_max = max;
  end for
...
 end for
end for
return (global_min, global_max);
```

Fig. 3. Interval evaluation using parameter domain splitting into subintervals

Affine operators always result in ranges expressed as a linear combination of noise parameters $\epsilon_i \in (-1, 1)$. Hence, affine transformations are able to exploit first order dependencies between ranges and sub-expression by performing a mixture of numeric and symbolic computations. However, second and third order dependencies between sub-expressions remain unexploited because of introducing new noise symbols, such as $\epsilon_3 = \epsilon_1 \times \epsilon_2$, for each multiplication and division operation.

For example, lets consider the expressions $E = x^2 + y^2 - 2xy$ with $x \in (3, 8)$ and $y \in (1, 2)$. The exact co-domain for expression E is $(1, 49)$. IA overestimates the co-domain as $(-22, 62)$. Applying AT, variables x and y are expressed as $x = 5.5 + 2.5 \times \epsilon_1$, and $y = 1.5 + 0.5 \times \epsilon_2$. Then, the co-domain of the expression E is $16 + 20\epsilon_1 - 4\epsilon_2 + 6.25\epsilon_3 + 0.25\epsilon_4 - 2.5\epsilon_5$, $\epsilon_i \in (-1, 1)$. The estimated range is $E \in (-17, 49)$. For this example, AT improves the upper bound of the range. The lower bound is largely overestimated, because of neglecting second order dependencies between noise symbols.

Definition: Simplified Affine Transformation (SAT) neglects the nonlinear term of an AT. SAT addition and subtraction are the same as those for AT. SAT multiplication operator is $I_1 \times I_2 = cv_1\ cv_2 + rad_2\ cv_1\ \epsilon_1 + rad_1\ cv_2\ \epsilon_2$. Division is calculated using SAT multiplication operator. The relative error (RE) of SAT multiplication is equal to $\frac{rad_1\ rad_2}{cv_1\ cv_2}$.

As compared to AT, SAT operators have the advantages of shorter evaluation time and improved precision. For AT, the total number of noise symbols is equal to the number of variables plus the number of multiplication and division operations. In contrast, SAT operators introduce a number of noise symbols equal to the number of parameters. Less noise symbols decrease the required evaluation time. Second, as the previous example shows, second and higher order dependencies can be significant for AT. The effect of dependencies can be diminished by splitting variable ranges into multiple subintervals [14]. This is infeasible for AT, as its computational complexity is already too large. As SAT requires much lesser noise symbols, domain splitting can be applied to mitigate the effects of second and higher order dependencies.

We argue that the accuracy of SAT multiplication and division can be improved by intelligently splitting the variable domains into multiple segments. The evaluation error converges to zero, as the number of variable domain segments increases [14]. Table 1 shows the ranges computed for expression $E = x^2 + y^2 - 2x\ y$, with $x \in (3, 8)$ and $y \in (1, 2)$. The exact co-domain for E is $(1, 49)$. Columns 3 and 4 indicate the number of segments used to split the range for variables x and y. Column 2 presents the number of subinterval combinations (equal to the product of columns 3 and 4). Columns 5 and 6 show the left and right limits for the interval ranges for expression E. The table shows that using more subintervals improves the accuracy of the result. For example, using 1,000,000 subintervals (row 6) offers a very accurate result, within 2% of the exact range.

	# nr segm.	# nr segm. for var. x	# nr segm. for var. y	left limit	right limit
1	4	2	2	-8.5	51
2	25	5	5	-2.62	49.98
3	100	10	10	-0.78	49.52
4	400	20	20	0.11	49.26
5	10,000	100	100	0.82	49.05
6	1,000,000	1,000	1,000	0.98	49.00
7	40	20	2	-0.76	48.6
8	25,000	500	50	0.92	48.97
9	40	2	20	-8.1	51.15
10	25,000	50	500	0.69	49.14

TABLE I

Experimental results for different multiplication sequences

Rows 7-10 motivate that a very good accuracy can result, if splitting is intelligently performed on the intervals. For 25,000 subinterval combinations (row 8) the final accuracy is still within 8%. Finally, with a small number of combination (40 subinterval pairs in row 7), the estimated range for E is significantly improved as compared to the range estimated through AT, which is the range (-17, 49) in this case. Femia *et al* [7] explain that, in general, the number and lengths of the splitting segments are correlated to the nonlinearity of the function. They show that finding the minimum number of splitting segments for a certain range accuracy is a hard problem, and suggest a genetic algorithm (GA) to solve it. This solution is not applicable to analog synthesis because of its computational complexity. Instead, we propose a constructive technique for deciding the number of segments a range has to be divided into, so that large ranges should be split into more subintervals than shorter ranges.

The range of parameter i has to be divided into n_i equal subintervals (either in linear scale or log scale). The challenge is to intelligently split the variable ranges knowing that the complexity of the range evaluation algorithm in Figure 3 increases according to $O(\prod^{\#variables} nr\_seg_i)$, the number of variables and nr_seg_i - the number of segments for dividing the domain of variable i. Assuming that the total number of segments is given (depending on how much complexity can be afforded), this number has to be distributed to variable domains, such that the resulting accuracy is maximized. For a product term, the number of segments is distributed proportionally to the domain width: $nr\_seg\ _i = \frac{total\ nr\ segments}{\sum_{\forall variables} var\ domain\ lengths} \times domain\ length\ of\ var_i$. Note that range splitting does not affect the number of noise symbols: the total number of noise symbols remains equal to the number of variables. After splitting, the original interval $I_I = (a_i, b_i) = cv_i + rad_i\epsilon_i$ is rewritten as $\bigcup_{k=1}^{n_i} Subi_k$, where $Subi_k = (a_i + (k-1)\ \frac{(b_i a_i)}{n_i}, a_I + k\ \frac{(b_I a_i)}{n_i}) = cv_{I,k} + rad_{I,n_i}\epsilon_i$. $cv_{I,k} = a_I + (k - 0.5)\ \frac{(b_I a_i)}{n_i}$, and $rad_{I,n_i} = \frac{(b_I a_i)}{2\ n_i}$.

V. Experiment

The experimental setup used two examples: a third-order elliptic filters and a fifth-order elliptic lowpass filter. Both IA-based pruning and the exploration algorithm had been coded in C/C++. The effectiveness of the parameter domain pruning was studied by analyzing synthesis convergence, quantity and quality of solutions, and corresponding execution time.

A. Third-order elliptic lowpass filter

The first synthesis experiment addressed the third-order elliptic lowpass filter shown in Figure 1 [13]. The goal was to find the capacitor values and transconductance parameters, so that the filter has a 3db bandwidth of 12Mhz and the magnitude error with respect to the ideal magnitude response is minimized. Used feasibility ranges were $[1.0e-15, 1.0e-11]$ for capacitors and $[1.0e-6, 1.0e-2]$ for transconductance gains.

	Best cost	Iteration # for best cost	Total # of points with cost < 200	Execution time (hrs)
SA	524	6,190	0	6.3
SAT + SA	5.77	3,640	14	8.2

TABLE II
Comparison of the synthesis strategies

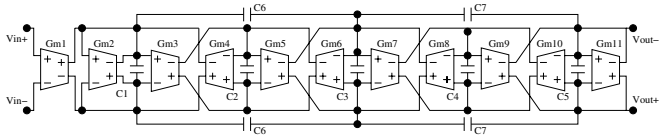

Fig. 4. **Fifth-order elliptic lowpass filter architecture**

We first run a synthesis experiment using Simulated Annealing (SA) algorithm with random starting point. The sampling step was fixed so that fifty samples were selected from each range (in a log scale). The best solution had a large cost value (524.1). The design point roughly met the bandwidth constraint (11.2Mhz), but failed to minimize the magnitude error to an acceptable range. So, it was considered that the solution did not represent a good quality design.

To observe the importance of the parameter pruning step based on interval arithmetic (Step 1 of the synthesis flow), we run simulated annealing for the pruned parameter domains. The cost of the best solution found drops significantly to 5.77. The reason is that SAT effectively narrowed down the variable ranges, so that the resolution of design points was better than before. We used magnitude response for parameter domain pruning. For example, after pruning, the value range for capacitor C_2 became $(1.0e-14, 1.0e-11)$, and the range for transconductor Gm_2 was $(0.000353, 0.005)$.

Table II compares the two different synthesis strategies. To study synthesis quality, we counted the number of found solutions with cost value less than 200 (representing good design points). Column 4 shows the number of good design solutions found in the parameter sub-domains. Column 2 shows the cost of the best solution identified during synthesis, and Column 3 indicates the iteration count at which the best solution was found. It can been seen that SAT-based parameter domain pruning followed by SA has much better convergence that SA only for the large parameter domains. Finally, Column 5 presents the synthesis time in hours. Compared to the case of SA only, SAT-based pruning followed by SA introduce a reasonable time overhead due to the computation time spent on pruning.

B. Fifth-order elliptic lowpass filter

The last experiment addressed a fifth-order elliptic lowpass filter used as a channel select lowpass filter in a Bluetooth receiver [16] [17]. The filter architecture is shown in Figure 4. Eighteen design variables were identified for this filter, nine for the capacitors with two matched pairs and eleven for the transconductor gains. Following the specifications, the goal was to synthesize the filter, so that it has the cutoff frequency of 400kHz, and notches at 1MHz and 3MHz to suppress adjacent channels, while maximizing image rejection and minimizing quality factor.

Table III presents synthesis results for SA and SAT followed by SA. Using the same feasibility ranges for design parameters as before, the best design synthesized with SA had a cost value of 1044.84, which does not represent a constraint satisfying design. In contrast, SAT and SA found a design point of cost 9.97. This point represents a filter with cutoff frequency of 400KHz, notches at 1MHz and 3MHz, image rejection of 85db, and quality factor of 1.26. In this experiment, we also used only magnitude response to prune parameter domains. In the exploration phase, all the related performance metrics are considered and evaluated from simulation data. Compared to SA only, IA and SAT increase the execution time by 1.7 times. In general, with the increase of the number of design variables, the time overhead for IA-based parameter domain pruning increases dramatically.

	Best cost	Iteration # for best cost	Total # of points with cost < 200	Execution time (hrs)
SA	1044	27,900	0	6.6
SAT + SA	9.97	6,922	9	11.2

TABLE III
Comparison of the synthesis strategies

VI. Conclusion

This paper describes a novel method for parameter domain pruning and its use for analog synthesis of filters. The goal is to identify parameter sub-domains that are more likely contain feasible and good design solutions without requiring knowing these parameter sub-domains as inputs. Instead, these parameter sub-domains has been automatically identified using the proposed Simplified Affine Transforms (SAT) operators. Then, pruned variable sub-domains are searched using exploration based algorithms. Experiments have shown that SAT followed by simulated annealing algorithm has a much better convergence than simulated annealing alone. Future works attempt to further improve the efficiency parameter domain pruning based on SAT, and extend the methodology to large circuits and systems with many design variables.

Acknowledgment

Supported by Defense Advanced Research Projects Agency (DARPA) and managed by the Sensors Directorate of the Air Force Research Laboratory, USAF, Wright-Patterson AFB, OH 45433-6543

References

[1] G. Gielen *et al*, "Computer Aided Design of Analog and Mixed-signal Integrated Circuits", *Proc. of IEEE*, vol 88, No 12, Dec 2000, pp1825-1852.
[2] M. Hershenson *et al*, "Optimal design of a CMOS op-amp via Geometric Programming", *IEEE Trans. on CADICS*, Vol.20, No.1, Jan 2001, pp.1-21.
[3] H. Tang *et al*, "Synthesis of Continuous-Time Filters and Analog to Digital Converters by Integrated Constraint Transformation, Floorplanning and Routing", *Proc. of Great Lakes Symposium on VLSI*, 2003, pp 207-210.
[4] H. Tang *et al*, "Towards High-Level Synthesis of Analog and Mixed-Signal Systems from VHDL-AMS Specifications - A Case Study for a Sigma-Delta Analog-Digital Converter", *Languages for System Specification*, Kluwer Academic Publishers.
[5] W. Kruiskamp *et al*, "DARWIN: CMOS opamp Synthesis by means of a Genetic Algorithm", *Proc. of Design Automation Conference*, 1995, pp. 433-438.
[6] A. Doboli *et al*, "Hierarchical Optimization for Synthesis of Linear Analog Systems", *Proc. of ISCAS*, 2001.
[7] N. Femia *et al*, "Genetic Optimization of Interval Arithmetic-Based Worst Case Circuit Tolerance Analysis", *IEEE Trans. on Circuits and Systems I*, Vol. 46, No. 12, Dec 1999, pp. 1441-1456.
[8] N. Femia *et al*, Ture Worst-Case Circuit Tolerance Analysis using Genetic Algorithms and Affine Arithmetic, *IEEE Trans. On Circuits and Systems I*, Vol. 47, No. 9, Sep 2000, pp. 1285-1296.
[9] L. Figueiredo *et al*, Fast Interval Branch-and-Bound Methods For Unconstrained Global Optimzation with Affine Arithemetic, *Technical Report IC-97-08, Institute of Computing, Univ. of Campinas*, June 1997.
[10] Lemke *et al*, Analog Circuit Sizing Based on Formal Methods Using Affine Arithmetic, *Proc. of Inter. Conf. on CAD*, 2002, pp486-489.
[11] C. Fang *et al*, Toward Efficient Static Analysis of Finite-Precision Effects in DPS Applications via Affine Arithmetic Modeling, *Proc. of Design Automation Conference*, 2003, pp.496-501.
[12] C. Grimm *et al*, Refinement of Mixed-Signal Systems with Affine Arithmetic, *Proc. Of Design, Automation and Test in Europe*, 2004.
[13] B. Ray *et al*, "Efficient Synthesis of OTA Networks for Linear Analog Functions", *IEEE Trans. on Computer Aided Design of ICS*, Vol. 21, May 2002, pp.517-533.
[14] R. Moore, "Interval Analysis", *Prentice Hall*, 1966.
[15] J. Stolfi *et al*, "Self-Validated Numerical Methods and Applications", *Proc. of 21st Brazilian Mathematics Colloquium*, Jul 1997.
[16] B. Pankiewicz *et al*, "Fifth Order Lowpass Continuous-Time CMOS OTA-C Filter", *Proc. International Conference on Signals and Electronic Systems*, 2002, pp.221-226.
[17] A. Zolfaghari *et al*, "A Noninvasive Channel Select Filter for a MOS Bluetooth Receiver", *Proc. of IEEE Custom Integrated Circuits Conference*, 2002.