LAYOUT-AWARE ANALOG SYNTHESIS BASED ON MULTI-LEVEL OPTIMIZATION, PERFORMANCE MODEL APPROXIMATION AND SOLUTION SPACE PRUNING

Hua Tang, Hui Zhang and Alex Doboli

Department of Electrical and Computer Engineering,
Stony Brook University (State University of New York), Stony Brook, NY, USA
Email: adoboli@ece.sunysb.edu.

Abstract: This paper presents a new multilevel optimization algorithm for layout-aware synthesis of analog systems. The synthesis methodology integrates (1) block parameter search, (2) block placement, and (3) global interconnect routing while maintaining an accurate perspective about layout parasitic. Multilevel optimization conducts a sequence of exploration steps, in which each step uses performance models of superior accuracy (with respect to the previous steps), while searching reduced parameter domains. The paper defines original algorithms for design parameter abstraction, abstraction error evaluation, and design space pruning. Theoretical and experimental results show a superior convergence of the multilevel algorithm ascompared to flat exploration. Execution time is also much improved without trading-off quality.

Keywords: electronic design automation, analog system synthesis, optimization.

1. INTRODUCTION

The need for mixed-signal designs is predicted to dramatically increase in the near future (Gielen, 2000). The digital part of mixed-signal systems can be efficiently designed with a low effort using modern high-level, logic-level, and physical-level design automation tools. In contrast, there is a lack of synthesis environments for analog circuits (Gielen, 2000). As a result, analog designs continue to seize a considerable portion of the design time of mixed-signal systems. There is a persistent need for synthesis tools to level design productivity and quality of analog circuits. This paper presents a novel method for analog system synthesis under close observation of layout parasitic.

Existing CAD methods for analog circuits mainly focus on layout generation (Cohn, 1991, Lampaert, 1999, Malavasi, 1996) and circuit sizing (Dhanwada, 1999, Krasnicki, 1999, Kruiskamp, 1995, Hershenson, 2001). Layout design tools follow an optimization-based place-and-route approach, in which layout is generated by an optimization process, e.g., simulated annealing, genetic algorithms etc. driven by a cost functions expressing criteria, such as performance degradation, total wire length, area, and so on (Cohn, 1991, Lampaert, 1999). However, it is possible that the fixed transistor sizes do not leave enough performance margins to accommodate layout-induced performance degradations. In this case, the final design is incorrect. Costly re-iterations through circuit sizing and layout generation are needed to get constraint satisfying designs.

Circuit sizing techniques find the transistor dimensions that optimize the performance requirements expressed by the designer (Krasnicki, 1999, Kruiskamp, 1995, Hershenson, 2001, Ochotta, 1996). Hershenson et al. (Hershenson, 2001) propose GPCAD environment, which optimally sizes transistors using geometric programming methods and performance models expressed as posynomial functions. MAELSTROM (Krasnicki, 1999), among others, employs genetic algorithms and SPICE simulation for transistor sizing. A main limitation of traditional circuit-sizing methods is the assumption of having minuscule layout parasitic, so that it can be disregarded. This is not, however, the case for many applications (Crols, 1995, Dessouky, 2001, Hastings,

2001, Vancoreland, 2001). Layout-induced parasitic, i.e. interconnect parasitic and device couplings through substrate, have a significant impact on performances of high-speed, high accuracy analog and RF circuits (Hastings, 2001). Performances of these circuits, after layout-parasitic extraction, might not meet the constraints. A tight integration of system synthesis and layout design is compulsory for having realistic estimations of layout parasitic.

A possible layout-aware synthesis approach would be to combine the sizing and layout design steps as part of the same exploration loop. This strategy is called *flat exploration*. The main difficulty of this methodology is its poor convergence due to the many variables that need to be simultaneously optimized over large domains of values. A typical example such as the 6-th order filter involves around 190 variables. Similar to other work (Dhanwada, 1999), our experiments with flat exploration methods showed poor convergence and long exploration times for a significant number of applications. Krasnicki et al. (Krasnicki, 1999) propose a distributed genetic algorithm to tackle the complexity of the solution space. While offering speed-up, this approach does not address the poor convergence of typical heuristic algorithms for problems with large number of variables.

This paper presents an original multilevel optimization algorithm for synthesis of analog systems. The algorithm performs combined block parameter search, block placement, and wire routing under strict observance of layout parasitic. To address the large complexity of the solution space, the algorithm conducts a sequence of optimization steps, where each step uses performance models of superior accuracies (with respect to the previous steps), while searching reduced parameter domains. Accuracy differs because distinct number of parameters is used in the models. The first optimization step starts with the coarsest performance models, thus having the least number of parameters. Based on the approximation error of the models, this step prunes unattractive parameter sub-spaces. The pruning criterion was defined such that eliminated subspaces do not include attractive design solutions, even if detailed models were used. Then, multilevel optimization proceeds with (1) gradually considering performance models with superior accuracies (thus, involving more parameters), while (2) searching parameter values located in the contracted domains. Domain contraction is due to pruning. To motivate the effectiveness of the method, the paper presents design examples for which flat exploration offered poor solutions, but which were successfully handled by multilevel optimization.

The theoretical core of the multilevel optimization algorithm is based on global nonlinear optimization (McCormick, 1983, Walshaw, 2001). The originality of the paper consists in refining the mathematical theory for the context of analog synthesis. We define criteria and algorithms for design parameter abstraction, abstraction error evaluation, and space pruning. The paper stresses on the theoretical aspects of the algorithm, as preserving optimality during multilevel optimization is critical. The convergence of the algorithm is superior to that of flat exploration. Execution time is also much improved.

Recently, there has been effort to combine transistor sizing with layout design. Vancorenland et al. (Vancorenland, 2001) discuss a layout-aware transistor sizing method for RF circuits. A layout template is assumed for a given circuit. Transistor dimensions are flexible in the template, but the placement and routing of the devices is already decided. Dessouky (Dessouky, 2001) proposes a similar approach. This technique is well suited for predefined circuits, like mixers. However, if a new application is targeted or a new circuit topology is contemplated, it will take a large designer effort to prepare the knowledge needed for synthesis. Our method is not limited to one application type as placement and global routing are part of synthesis.

This paper is organized as follows. Section 2 motivates the proposed multilevel optimization algorithm. Section 3 details the theoretical basis for the method. Section 4 introduces the synthesis methodology and algorithms. Experiments are shown in Section 5. Finally, Section 6 provides conclusions.

II. MOTIVATION

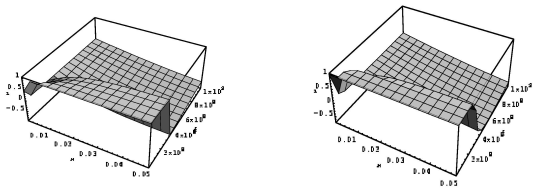

Fig.1 Variables with dominant and reduced influence

The main practical difficulty of synthesis through flat optimization is the large number of variables that need to be simultaneously explored over extensive domains of values. For example, designing a 6-th order filter up to the layout involves optimizing around 196 real-domain variables (48 variables describe OpAmp constraints, 6 variables are for capacitors, 30 variables are for resistors, 48 variables express placement, and 64 variables are for routing). From the point-of-view of an optimization algorithm, the large number of variables increases the size of the neighborhood sets, as many incremental changes can be defined for each iteration. Secondly, many exploration steps need to be performed to reach an optimal solution from a given solution point. This decreases the rate of convergence of the exploration algorithm. In fact, it is very likely that many of the local optima are never reached during the search.

Parallel and multilevel exploration methods are two orthogonal ways to tackle these difficulties.

Parallel optimization algorithms divide the variable domains into subsets, and allocate each solution space region to a different processor (Reeves, 1993). Assuming that there exists true data parallelism for the synthesis problem, a theoretical speed-up of *m* is achieved by using *m* processors as compared with flat optimization. Real speed-up is smaller, as data (local optima) need to be constantly exchanged between processors. While offering speed-up, the main limitation of parallel optimization algorithms is the extra hardware and software cost they require.

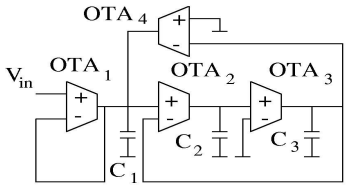

Fig. 2 Third-order elliptic filter

Multilevel optimization addresses the problem of solution space complexity by distinguishing the search variables based on their impact on the optimization and constraint functions. Using the transfer function of a system or designer knowledge, variables are grouped into (1) *variables with dominant influence* and (2) *variables with reduced influence*. Sensitivity analysis can be used to identify variables with dominant influence.

For the elliptic filter in Figure 2 (Ray, 2002), Figure 1 shows the variables with dominant and reduced influence. The transfer function of the filter is

$H(s)=[gm_1gm_2gm_3-gm_1C_2C_3\varpi^2]/[gm_1gm_2gm_3+ gm_2gm_3gm_4 +gm_1C_2C3\ \varpi^2+ (gm_2gm_3C_{-1} - C_1C_2C_3\ \varpi^2)\ s]$ (Ray, 2002).

The left graph in Figure 1 presents the filter amplitude as a function of frequency and gain gm_1. The right graph in Figure 1 shows the filter amplitude as a function of frequency and gain gm_1, if the transfer function is approximated to $H'(s) = (gm_1) / (gm_1 + C_1\ s)$. This is a reasonable approximation for the frequency range $f \in (1,10^8)$ Hz. Note that function $H'(s)$ preserves the shape of the original function $H(s)$, such that optima are located in the same region. Thus, variables gm_1 and C_1 have a dominant influence.

Multilevel optimization concentrates first on addressing the variables with dominant influence. The number of searched variables is significantly reduced as compared to flat optimization, because variables with reduced influence are abstracted. The exploration process isolates regions, which are attractive (e.g., they contain good local optima), and prune non-competitive regions (e.g., their local optima are much worse then the optima of other regions). Then, abstracted variables are gradually introduced-back to the exploration space. However, the exploration algorithm focuses only on the attractive regions, thus, on a smaller space. This strategy improves the convergence of the optimization without perturbing the quality of the search. For the discussed example, multilevel optimization first explores the dominant variables gm_1 and C_1 over their entire domain to identify the space regions where the filter response is satisfactory. For example, the region for $gm_1 < 0.3$ μA/V is pruned as the cost is high. Then, reduced influence variables are gradually re-introduced into the estimation model, while exploration concentrates on the attractive regions found during previous exploration steps. Through intelligent search, multilevel optimization speeds up execution and improves convergence without using more hardware and software resources than flat exploration.

III. MULTILEVEL OPTIMIZATION

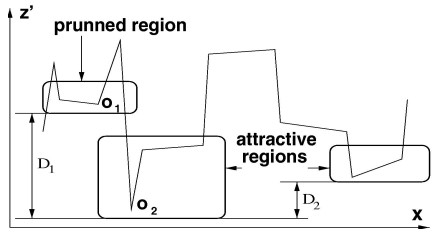

Fig. 3 Multilevel optimization

Using the concept of dominant and reduced influence variables, Figure 3 offers an intuitive description of multilevel exploration. Lets assume that the function $z = f(x, y)$ is to be optimized. Assuming that variable y has a reduced influence for a tolerable error, an approximation $z' = f'(x)$ of function f is obtained by abstracting variable y. Figure 3 shows the curve of the abstracted function f'. Function f' is used to identify *attractive regions* of the solution space, and prune less promising regions, as shown in Figure 3.

A subsequent exploration step will explore for the optima of the initial function f by concentrating the search on the attractive regions found during the first step. This section discusses the theoretical foundation of the multilevel optimization algorithm.

3.1 Dominant and reduced influence variables, approximation errors, attractive and pruned regions

Definition: Variable $x_i \in (x^{min}_i, x^{max}_i)$ has a reduced influence on function $F(x_1, \ldots, x_n)$ with respect to the approximation error $\varepsilon > 0$, if for any constant $x^c_i \in (x^{min}_i, x^{max}_i)$, the relationship $\|F(x_1,x_2,\ldots, x_i, \ldots, x_n) - F(x_1,x_2,\ldots, x^{\wedge c}_i, \ldots, x_n)\| < \varepsilon$ holds. Variable x_i has a dominant influence on $F(x_1, \ldots, x_n)$ with respect to error $\varepsilon > 0$, if it has not a reduced influence.

For the filter in Figure 2, variables gm_1 and C_1 have a dominant influence, and variables gm_2, gm_3, gm_4, C_2, C_3, and the layout parasitic have a reduced influence.

The error range introduced by eliminating a variable from function F, can be estimated based on the function and the variable domains of F. We assume that function $F(x_1,...,x_n)$ has bounded first order derivates ($\partial F/\partial x_i \in (L^{min}_i, L^{max}_i)$, and $x_i \in (x^{min}_i, x^{max}_i)$, which denotes its *feasibility range*. Values $x^{min}_i, x^{max}_i > 0$, because they are physical dimensions.

Lemma: The error ε introduced by eliminating variable x_i from function F is $\varepsilon \in$ (**min** $\{L^{min}_i \times x^{min}_i, L^{min}_i \times x^{max}_i\}$, **max** $\{L^{max}_i \times x^{max}_i, L^{max}_i \times x^{min}_i\}$).

Proof: The elimination of x_i assumes that $\partial F / \partial x_i$ in the Taylor series of F disappears, which results in the mentioned approximation error. The lemma considers the cases where bounds L^{min}_i and L^{max}_i can be both positive and negative.

Definition: For a given a permutation $\sigma(m)$ of m numbers in the set $\{1, ..., n\}$, $F^{\sigma(m)}(x_1,...,x_n)$ is the function obtained through eliminating variables x_i, where indexes i are defined and appear in the same order as the integers in the permutation $\sigma(m)$.

For example, if $\sigma(3)=\{3, 1, 4\}$ then $F^{\sigma(3)}(x_1,...,x_5)$ is the function obtained after eliminating x_3, x_1 and x_4.

Lemma: For a given a permutation $\sigma(m)$, the error of the corresponding abstraction is $\varepsilon = \| F^{\sigma(m)}(x_1,...,x_n) - F(x_1,...,x_n) \| \in (\sum_{\sigma(m)}$ **min** $\{L^{min}_i \times x^{min}_i, L^{min}_i \times x^{max}_i\}$, $\sum_{\sigma(m)}$ **max** $\{L^{max}_i \times x^{max}_i, L^{max}_i \times x^{min}_i\})$.

Proof: Proof is based on the cumulative effect of the errors introduced by each variable abstraction.

Lemma: Let $\varepsilon_1 \in (L^{min}, L^{max})$ be the approximation error of $F^{\sigma(m)}$. Point o_1 is assumed to be the *unique* local minimum of $F^{\sigma(m)}$ for the subspace R_1. Point o_2 is assumed to be the *unique* local minimum of $F^{\sigma(m)}$ for the subspace R_2. Without loosing optimality, subspace R_1 can be pruned from the search space, if $o_2 < o_1$, and $o_1 + L^{min} > o_2 + L^{max}$.

Proof: Refer to Figure 3 for an illustrative example. The intuition behind the proof is that point o_1 is worse than point o_2 by a margin greater than the maximum error introduced through the abstraction. Thus, point o_1 cannot become better than point o_2 under any circumstances.

This lemma is very important for stating the conditions under which space pruning can be conducted without affecting optimality.

3.2 Multilevel optimization through variable approximation and space pruning

Figure 4 presents the pseudo-code of the multilevel optimization algorithm. The multilevel synthesis strategy can be defined for any number of levels. The strategy starts from a detailed optimization and constraint functions and the complete solution space SP. Then, the technique classifies free variables into those with dominant and reduced influence (lines 2-4). A sequence of variable abstractions is defined after this step (line 5).

The second part conducts a sequence of explorations. It starts with the most comprehensive abstraction (line 6), identifies attractive regions, and prunes non-optimal zones. Lets assume that subspaces R_1 and R_2 are isolated during the search, each of them containing one local optima (thus the function is convex over R_i). Point o_i is the local optimum for region R_i, i=1,2. If $o_1 + L^{min} > o_2 + L^{max}$, subspace R_1 can be pruned without altering the optimality of the problem. Range (L^{min}, L^{max}) is the approximation error of the current performance model. Region R_2 is an attractive region, and region R_1 is a pruned region. By extending the reasoning for the entire solution space, the strategy isolates a set of attractive regions (line 9). Using the sequence of variable abstractions, the algorithm identifies the next refinement in the sequence (lines 10-11). The exploration process is repeated using the refined function defined over the set of attractive regions (line 8). The last iteration performs an exploration of the attractive regions using the non-approximated model.

```
PROCEDURE multi_level optimization IS
INPUT:
   F – function to be optimized
   D_i – domain of free variable x_i
   ε – maximum approximation error
BEGIN
(1) SP = D_1 U D_2 U ... U D_n;
(2) for all free variables x_i do
(3)   evaluate the approximation error
       introduced by abstracting x_i in F;
     end for
(4) identify the set S of variables x_i
      that can be abstracted so that the
      resulting total error < ε;
(5) order variables x_i in set S in
      decreasing order of their dominance;
(6) F' = F^σ(i | xi∈S)(x_1, ..., x_n);
(7) AR = SP;
(8) for all x_i ∈ S, in their order in S do
(9)    AR = identify the attractive sub-spaces of AR by exploring
             F' and pruning unattractive subspaces;
(10)   S = S - x_i;
(11)   F' = F^σ(i | xi ∈ S)(x1, ..., xn);
     end for
END PROCEDURE
```

Fig.4 Multilevel optimization method

Lemma: The multilevel optimization algorithm finds the global optimum, if pruned regions are convex.

Proof: As pruned regions are convex, they have a single local optimum. Optimality is preserved due to the lemma in Section 3.1.

3.3 Discussion

The speed-up of multilevel optimization with respect to flat optimization depends mainly on the number of variables that are approximated, and less on the

solution space reduction through pruning. The motivation assumes that variables in function $F(x_1, ..., x_n)$ can be abstracted through a set of m steps, each step abstracting k variables. Also, lets assume that each iteration decreases the size of the solution space by a factor of q, $q > 1$. Finally, the complexity of the algorithm to identify the local optima over a space of dimension d is assumed to be of the order $\boldsymbol{O}(d^p)$. Lets assume that the most abstract function has n variables (the first abstraction in the sequence). Finally, we assume that each variable domain has the size l.

A flat exploration process has complexity $CP_1 = \boldsymbol{O}((n + k \times m \times l)^p)$. The multilevel exploration process has complexity $CP_2 = \boldsymbol{O}((n \times l)^p) + \boldsymbol{O}(((n + k) \times l/q)^p) + (((n + 2 \times k) \times l/q^2)^p) + ... + \boldsymbol{O}(((n + m \times k) \times l/q^m)^p)$. The inverse of the speedup is $CP_2/CP_1 = \sum_{i=0}^{m} ((n + i \times k)/(n + m \times k))^p \times 1/q^{ip}$. The first term has the most dominant influence on the speed-up. Thus, to maximize speed-up the first term should be minimized, which corresponds to aggressive variable abstraction rather than forceful domain reduction.

We considered a set of numerical examples to evaluate the resulting speed-ups. In the initial case, n=40, m=3, k=3, p=5, q=4/3. This example corresponds to the second order filter example. Local optima are identified with a polynomial algorithm of complexity $\boldsymbol{O}(n^5)$ (reasonable for an optimization heuristics like tabu-search). The estimated speed-up through multilevel optimization is around 1.8. If a more aggressive parameter abstraction procedure were to be used, such as n=20, m=4, k=5, p=5, q= 4/3, speed-up increases to about 13. If the pruning of less attractive regions is more efficient, e.g., n=40, m=3, k=3, p=5, q=2, then speed-up is around 2.56. This offers a numerical explanation for the statement.

This discussion suggests that multilevel optimization tends to be more effective in space exploration than parallel implementations. Parallel implementations realize a decreasing of the factor l, which is not part of the speed-up formula. Nevertheless, smaller size spaces tend to increase the value of q. The impact on speed-up is probable less than for parallel implementations. A parallel realization of the method in Figure 4 further speeds-up optimization.

IV. SYNTHESIS METHODOLOGY

Figure 5 illustrates the proposed layout-aware synthesis method. The methodology is an instantiation of the multilevel algorithm in Figure 4. It performs multilevel exploration for integrated (1) block parameter search, (2) block placement, and (3) global signal routing. System architectures are given as inputs. Architectures are netlists of active circuits i.e. OpAmps, OTA etc. and passive elements like resistors and capacitors. Figure 2 shows a sample architecture. The output consists of a sized, placed and routed architecture that realizes the required AC and transient behavior (like in Figure 10).

The methodology debuts with the step of finding dominant and reduced influence variables. This corresponds to lines 2-5 in Figure 4. The step *calculates symbolic expressions* for the performance attributes of a system. The models incorporate parameters such as OpAmp (OTA) gains, dominant poles, input/output impedances, resistor and capacitor values, and interconnect parasitic. Compact linear symbolic models are built using the method presented (Doboli, 2001). A feasibility range describes each of the architecture parameters. The *variable abstraction step* uses the symbolic models and feasibility ranges to calculate the error introduced by eliminating each variable. Then, variables are clustered, so that each cluster results in the same approximation error. The error per cluster is an input parameter to the algorithm. The variable clustering method considers parameters in the increasing order of the introduced error. Thus, initial clusters will include more variables than the latter. As motivated by the discussion in Section 3.3, this strategy offers the best speed-up for the multilevel optimization, because it allows quick pruning of unattractive subspaces. Subsection 4.1 more details on the variable abstraction method.

The next part instantiates lines 6-11 in Figure 4 for layout-aware analog synthesis. The initial exploration is conducted using the maximum variable elimination, thus models with highest approximation errors. Successive exploration re-introduces variables according to the clustering found by the variable abstraction step. The actual exploration algorithm is based on the tabu-search method (Reeves, 1993). Subsection 4.2 discusses the design steps performed by tabu-search.

Using layout parasitic modeling (Hastings, 2001), and technology-dependent values for the parasitic resistance and capacitance per unit length, a SPICE description of the circuits is generated. This circuit includes layout parasitic. AC and transient performances are obtained for each solution through SPICE simulation. Performances are used to calculate a cost function C, which controls the exploration algorithm for synthesis.

$$C = \sum_i \alpha_i \times ||(\text{Performance}^{\text{obtained}}{}_i - \text{Performance}^{\text{desired}}{}_i) / \text{Performance}^{\text{desired}}{}_i|| + \beta \times \text{Area}_{\text{system}}$$

Cost function C is the weighted sum of the relative error for performance parameters (i.e. DC Gain, 3db Bandwidth etc.) and the silicon area of the design. Each exploration level identifies attractive exploration regions and prunes unappealing sub-spaces. Subsection 4.3 discusses this task. Attractive regions are used by the subsequent explorations that involve more design parameters.

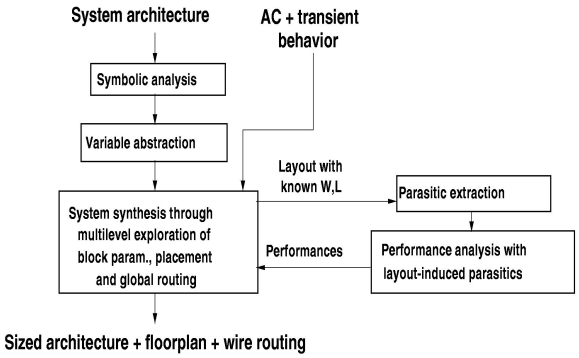

Fig. 5 System synthesis methodology

4.1. Identification of Dominant and Reduced Influence Variables Through Abstraction

The variable abstraction sequence is constructed in a greedy way. First, the method calculates for each variable the error resulting, if the variable is eliminated from the model. The variable that offers the smallest error is introduced first into the abstraction sequence. This variable has a reduced influence. An abstraction of the initial function is obtained after eliminating the variable. Then, this function abstraction is used to identify the variable with the next smallest error. A new abstraction is built after eliminating this variable. The algorithm ends when the resulting error higher than a specified error. Variables that were not abstracted have a dominant influence. The abstraction sequence is used to cluster the variables, so that each cluster results in the same approximation error. Note that the greedy strategy balances between an aggressive variable elimination and the effectiveness of solution space pruning. More aggressive abstraction increases the error range, thus, reduces the possibility for pruning. However, as shown in Section 3, variable abstraction offers better speed-ups than pruning.

The approximation error calculation finds the error resulting after eliminating variable x_i from the optimization and constraint functions of a system. The error is bounded considering that the function variables are bounded too. We propose an approximation error calculation scheme, which uses the symbolic expressions of the system performances and constraints. Symbolic expressions are used for range calculation through operations on intervals.

4.2. Design Exploration

At each iteration, the tabu-search algorithm explores three design steps that are shown in Figure 6:

- *Block parameter search* is achieved by changing dimensions h and w of the active part of the layout tiles for a block. This step is presented in Figure 6(a). For example, the values of resistances and capacitances change to improve the AC performances of a filter, like its 3dB bandwidth. For resistors, resistances depended on the area according to the formula $R_{seg} = \rho l/(wt)$. For capacitors, capacitances depend by the expression $C_{seg} = \varepsilon_{ox} wl/t_{ox}$. For OpAmps, we assumed a fixed but sufficient DC gain of 60dB and a fixed slew-rate (SR) of 1.6 V/μs. Following model links the UGF of an OpAmp to its area $Area_{OpAmp}=171.91\times UGF^{0.2875}\times SR^{0.1688}$.
- *Block placement* is addressed by moving one block or swapping two blocks. As a result, the x and y coordinates of the tile corner points for the involved blocks are changed to reflect the new situation. Figure 6(b) shows the swapping of Block 1 and Block 2. The produced overlapping is solved as shown in the right part of Figure 6(b). Affected nets are re-routed after each change of block positions.
- *Global routing* is the process of finding an ordered sequence of channels, so that source and target terminals are linked together. The algorithm randomly picks a channel P in the sequence. The subsequence from the source terminal to channel P remains the same. The sequence from P to the target pin changes to include a different global route. Block terminals are assumed to be fixed on a given side of their rectangle layout block. Exploring for good terminal positions will extend our current work.

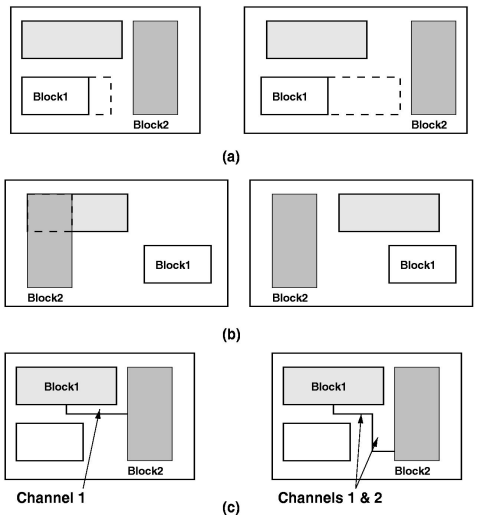

Fig. 6 Layout-aware block parameter exploration, placement, and global routing

4.3. Attractive Sub-space Identification. Pruning

Section 3.2 motivates that the global optimality of multilevel exploration is preserved, if pruned sub-spaces are convex. Establishing the convexity of a subspace is, however, difficult (McCormick, 1983). Instead, we approximate the convexity subspace identification problem by the condition that the sampled points form a convex envelope. This approximation criterion does not mathematically guarantee the convexity of the subspace, thus, other local optima might be located between the sampled points. This section offers a criterion to estimate the

amount by which the sampled optimum might differ from the real local optimum.

Figure 7 shows the subspace identification procedure. The sequence of points x_0, x_1, ..., x_7, x_8 corresponds to a convex envelope, and is detected during tabu-search exploration. Lets assume that x_0 is the current point under investigation. Tabu-search attempts at each step at changing each of the parameters (variables a and b in Figure 7). Point x_1 offers the best improvement of the cost function, if only one of the variables is changed. Similarly, the sequence of points x_2, x_3 is produced until reaching the point x_4, which acts as a local optimum for the sampled points. The sequence of points x_5, x_6, x_7, and x_8 corresponds to the increasing slope of the convex region, generated during tabu aspiration (Reeves, 1993).

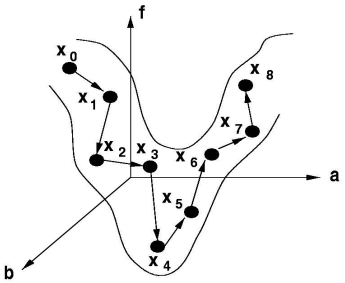

Fig. 7 Sub-space identification

The subspace represented by the sampled points x_0, x_1, ..., x_7, x_8 is described by the product $SSP = (a^{min}, a^{max}) \times (b^{min}, b^{max})$, where a^{min} = **min** ($x_0.a$, ..., $x_8.a$), a^{max} = **max** ($x_0.a$, ..., $x_8.a$), b^{min} = **min** ($x_0.b$, ..., $x_8.b$), and b^{max} = **max** ($x_0.b$, ..., $x_{8.b}$). $x_i.a$ indicates parameter a of the point x_i, and $x_i.b$ is parameter b of the point. The cost function of point x_4 approximates the local optimum of the subspace SSP. The procedure can be easily generalized for n parameters.

Lemma: For the subspace SSP, let f_o be the local optimum and f_a the optimum found through the presented sampling method. Let M be the maximum value of the derivate of the cost function, and ε the approximation error due variable elimination. Then, $\| f_o - f_a \| < \varepsilon$, if Δparameter = ε/M.

Proof: The formula results from the Taylor expansion of f. Intuitively, the lemma states that the parameters need to be varied with small steps, if the function has large variations, and with large steps, if the function has reduced variations. Under these conditions, the local optimum will be within a radius ε from the optimum point f_a found by the sampling process.

V.EXPERIMENTS

Experiments observed the effectiveness of our method in synthesizing high-frequency systems. The results were analyzed based on the quality of AC responses and layout compactness. The accuracy of the parasitic modeling was also considered. A third order elliptic filter and a sixth order low-pass filter are shown in the paper.

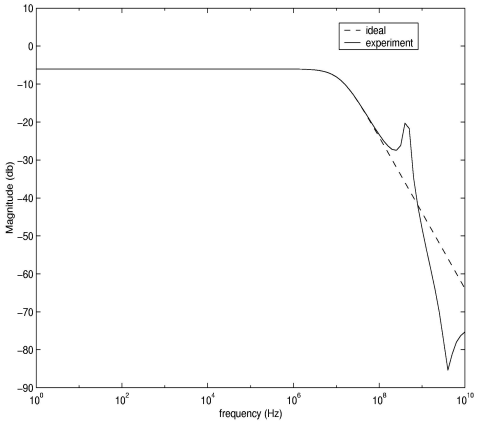

Fig. 8 AC response of the 12MHz elliptic filter

Experiments were run on a SUN 80 workstation. The optimization of the 3-rd order filter needed about 300 iterations and 3 hours. The 6-th order filter required about 1000 iterations and 20 hours.

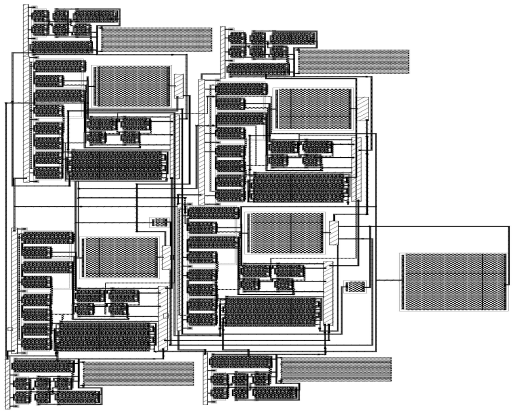
Fig. 9 3rd order elliptic filter layout

The third order elliptic low-pass filter was synthesized for a 3-dB bandwidth at 12 MHz. Following values resulted for the components: C_1=9.9pF, C_2=90fF, C_3=180fF, gm_1=408 μA/V, gm_2=387μA/V, and gm_3=400μA/V. Parasitic capacitances of wires have the same range of values as capacitors C_2 and C_3. This motivates the need for a combined sizing, placement, and routing. Figure 8 presents the frequency response of the filter. For the specified bandwidth, note the good resemblance between the specified and obtained AC response. The response was worse for the filter designed through a traditional method, which separated component sizing, placement, and routing. The 3dB point was shifted by about 3MHz. This indicates the impact of layout parasitic on the performance of the design, and thus, the need for contemplating parasitic and coupling effects during synthesis. Figure 9 shows the layout of the filter. The layout is fairly compact

The convergence for flat exploration was poor. The cost function oscillated without finding reasonable solutions. In our experiment, after 120 hours, the flat synthesis was still not converging. The proposed

multilevel algorithm had a far better convergence. The first optimization step explored only parameters gm_1 and C_1, as they have a dominant influence for the approximation error $\varepsilon < 10\%$. The range gm_1 = (380uA/V, 420uA/0V) and C_1 = (8pF, 12pF) was identified as an attractive region. The rest of the space for the gm_1 and C_1 values was pruned. The second optimization step used the complete model of the filter, including layout parasitic. The search concentrated only on the attractive regions for gm_1 and C_1. In this case, the algorithm converged quite rapidly to the solution with the response in Figure 8.

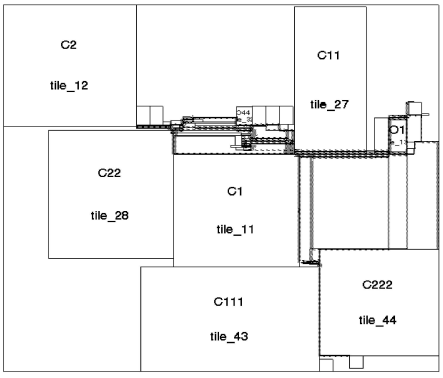

Fig. 10 Layout of the 6th order filter

Figure 10 shows the layout obtained for the 6-th order filter, the larger example. The AC response of the filter synthesized without considering parasitic capacitances had its 3dB point shifted by about 100kHz. This was clearly different from the required response, which was set at 300kHz. AC response was very much improved when considering the layout parasitic during synthesis. Algorithm convergence was better than for flat optimization. Finally, Figure 11 shows the signal-to-noise ratio (SNR) and dynamic range (DR) plots of the second-order ΣΔ ADC synthesized with the layout aware method. Maximum SNR is about 49 dB and DR is 64 dB.

VI. CONCLUSIONS AND FUTURE WORK

This paper presents a new multilevel optimization algorithm for layout-aware synthesis of analog systems. The synthesis method integrates block parameter search, block placement, and global interconnect routing, while maintaining an accurate perspective on layout parasitic. Multilevel optimization conducts a sequence of exploration steps, in which each step uses performance models of superior accuracy, while searching reduced parameter domains. The foundation of the algorithm is global non-linear optimization. Using a rigorous motivation, the paper defines original techniques for parameter abstraction, abstraction error evaluation, and design space pruning.

Experiments motivate the generality of the method, as it can be used for synthesis of different application types, like filters and ADCs. Produced designs offer good performance quality and layouts are compact. Most importantly, multilevel optimization offers a far better convergence than traditional, flat optimization.

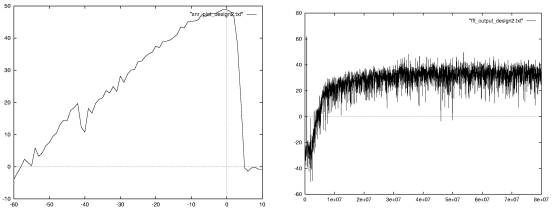

Fig. 11 SNR and DR for the second-order ΣΔ converter

REFERENCES

Cohn, J., et al. (1991). KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing, *IEEE JSSC*, Vol. **26**, pp. 330-342.

Crols, J., et al. (1995). A High-level Design and Optimization Tool for Analog RF Receiver Front-Ends, *Proc. of ICCAD*, pp. 550-553.

Dessouky, M. (2001). Design for Reuse of Analog Circuits, Ph.D. Thesis, *Univ. of Paris VI*.

Dhanwada, N., et al. (1999). Hierarchical Constraint Transformation using Directed Interval Search for Analog System Synthesis, *Proc. DATE*, pp. 328-335.

Doboli, A., Vemuri, R. (2001). A Regularity-based Hierarchical Symbolic Analysis Method for Large-scale Analog Networks, IEEE Trans. Circuits & Systems Part - II, Vol. **48**, No. 11, pp.1054-1067.

Gielen, G., Rutenbar R. (2000). Computer Aided Design of Analog and Mixed-signal Integrated Circuits, *Proc. of IEEE*, Vol. **88**, No. 12, pp. 1825-1852.

Hastings, A., The Art of Analog Layout, *Prentice Hall*, 2001.

Hershenson, M., et al. (2001). Optimal design of a CMOS op-amp via Geometric Programming, *IEEE Trans. CADICS*, Vol.**20**, No.1, pp. 1-21.

Krasnicki, M., et al. (1999), MAELSTROM: Efficient Simulation-Based Synthesis for Custom Analog Cells, *Proc. DAC*, pp. 945-950.

Kruiskamp, W., Leenaerts, D. (1995). DARWIN: CMOS op-amp synthesis by means of a genetic algorithm, *Proc. DAC*, pp. 433-438.

Lampaert, K., Gielen, G., Sansen W. (1999). Analog Layout Generation for Performance and Manufacturability, *Kluwer Academic Publishers*.

Malavasi, E., et al. (1996), Automation of IC layout with Analog Constraints, *Trans. of CADICS*, Vol. **15**, No. 12, pp. 1518-1524.

McCormick, G. (1983). Nonlinear Programming, *J. Wiley*.

Ochotta, E., et al. (1996), Synthesis of high-performance analog circuits in ASTRX/OBLX, *IEEE Trans. CADICS*, Vol. **15**, pp.273-294.

Ray, R., et al. (2002). Efficient Synthesis of OTA Network for Linear Analog Functions, *IEEE Trans. CAD*, Vol. **21**, No. 5, pp. 517-533.

Reeves, C. (1993). Modern Heuristic Techniques for Combinatorial Problems, *J. Wiley*.

Vancorenland, P., et al. (2001), A Layout-Aware Synthesis Methodology for RF Circuits, *Proc. ICCAD*, pp. 358-362.

Walshaw, C. (2001), Multilevel Refinement for Combinatorial Optimization Problems, *Mathematics Research Report*, University of Greenwich, 01/IM/73.