Accurate Performance Estimation using Circuit Matrix Models in Analog Circuit Synthesis

Almitra Pradhan and Ranga Vemuri

Department of ECE, University of Cincinnati, Cincinnati, OH 45221, USA
pradhaa@ececs.uc.edu, ranga@ececs.uc.edu

Abstract. Optimization based sizing methods allow automating the synthesis of analog circuits. Automated analog circuit synthesis techniques depend on fast and reliable estimation of circuit performance. This paper presents a highly accurate method of estimating performances by constructing models of the circuit matrix instead of the traditionally used performance models. Device matching in analog circuits is utilized to identify identical elements in the circuit matrix and reduce the number of elements to be modeled. Experiments conducted on benchmark circuits demonstrate the effectiveness of the method in achieving correct performance prediction. Results show that the performances can be predicted within a mean error of 0.1% compared to a SPICE simulation. Techniques such as hashing and near neighbor searches are proposed to expedite the matrix model evaluation procedure. These techniques avoid recomputations by saving previously visited solutions. The procedure is used for synthesizing analog circuits from various specifications such as performance parameters, frequency response. The proposed method gives accurate results for synthesis for various types of circuit specifications.

1 Introduction

Fast and accurate sizing of analog circuits has been a challenging problem in the EDA industry. Circuit sizing is the process of determining device dimensions and biasing of a given topology to achieve the desired performance goals.Automated synthesis methods are either knowledge based or optimization based. The former rely on expert knowledge for generating automated design plans or design equations. The latter methods on the other hand, construct sizing as a weighted cost minimization problem. Design variables (v_1,.., v_m) including device lengths and widths, biasing sources are identified for the circuit being sized. The design variable values that minimize the weighted performance cost is accepted as the sizing solution. Thus, the sizing problem can be formulated as follows:

$$minimize \sum_{i=1}^{N} Weight[i] * (Perf[i] - Perf_{spec}[i]) \tag{1}$$

where, Perf = F(v_1, ..., v_m)

An optimization algorithm such as Simulated Annealing (SA) or Genetic Algorithm (GA) proposes device sizes and bias from a search range and the evaluator verifies if the performance goal is met. The evaluator has to be both fast and accurate. Spice simulation, symbolic analysis and regression models have been used by researchers for performance evaluation. Spice simulation is the most accurate but requires a large runtime. Using symbolic analysis provides a faster alternative but suffers from term explosion for larger circuits. With macromodels, the relation between the design variables and circuit performance is captured by a black box abstraction. These evaluate much faster than direct simulation and can achieve speedy synthesis. However, the performance parameters are extremely difficult to model. Macromodels can suffer from inaccuracies and research efforts are directed at using complex modeling strategies to achieve good accuracy.

2 Related Work

This section reviews some of the methods proposed for optimization based sizing of analog circuits in recent years. Krasnicki *et al.* [1, 2] propose a sizing flow with a SPICE level simulator for predicting the circuit performance. Although, using a simulator gives highly accurate results, it proves expensive in terms of runtime. Symbolic analyzers used for performance prediction are also highly accurate as well as faster than spice [3, 4]. However, the limitation of symbolic models is the exponential increase of symbolic terms with circuit size making them less scalable.

Performance macromodeling has emerged as faster sizing technique compared to exact spice-like optimization approaches. Here, data for some chosen performance parameters is gathered at a number of sample points in the circuit design space. Regression models are then developed for each performance parameter. During sizing, these fast evaluating regression models are used instead of simulation to speed up the synthesis process. Wolfe *et al.* [5], Doboli *et al.* [6] used neural networks for regressing over performance parameters. Support Vector Machines were used for the same by Kiely *et al.* [7], Bernanandinis *et al.* [8] and Ding *et al.* [9]. Other techniques used for modeling include adaptive splines [10, 11], boosted regressors [12]. A review of several performance modeling methods proposed in recent years can be found in [13, 14].

3 Introduction to Circuit Matrix Models

To obtain performance parameters of an analog circuit at a given point in the search space, the system matrix of the circuit is generated. This matrix, also called the circuit matrix, is derived based on the Modified Nodal Analysis (MNA) formulation. The circuit matrix can be represented as follows:

$$(G + sC)x = B;$$
$$y = L^T x$$

here, G: conductance submatrix, C: susceptance submatrix, B: input vector, L: output vector, x: unknown state vector, y: output vector

Pre-defined MNA stamps for all circuit elements allow circuit matrix generation to be quite straightforward. The MNA stamp of a mosfet is written in terms of its small signal values such as transconductance (gm), output conductance (gds), capacitance (cgs, cgd, cgb) etc., whereas for other circuit elements stamps are in terms of the component values. The small signal values of mosfets are obtained by linearizing the circuit around the operating point. The circuit matrix is solved to obtain the frequency response of the circuit. Performance parameters such as the low frequency gain, Unity gain frequency (UGF), Gain Margin (GM), Phase Margin (PM) are calculated from the frequency response. In simulation based synthesis, the spice engine generates and solves the circuit matrix. Macromodeling approaches use fast evaluating models and eliminate the use of spice. As shown in fig. 1 macromodeling is possible at *two* places in the synthesis flow:

1. Modeling the performance parameters
2. Modeling the circuit matrix

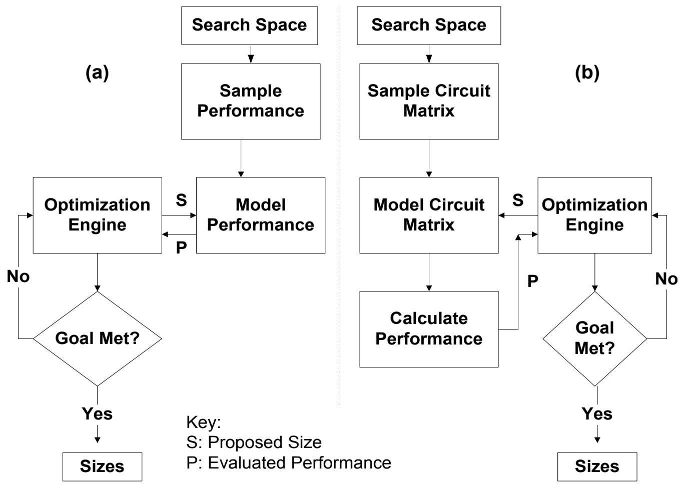

Fig. 1. (a) Performance Modeling Approach (b) Matrix Modeling Approach

Most of the existing macromodeling techniques use the first approach i.e. they model the performance parameters directly. Such methods greatly concentrate on the performance estimation speed, but suffer a tradeoff with accuracy. This paper presents an *alternative method* of estimating performance characteristics of linear analog circuits by constructing a model of the circuit matrix. The advantage, as will be seen, is that the matrix can be very accurately modeled even with simpler modeling approaches such as multivariate polynomial regression. Since it is possible to accurately estimate performance values, true design convergence is obtained by this method.

Performance is not directly modeled but it is calculated from the matrix model. Although this requires some extra computation time, the speed loss is not significant and is offset by the gain in accuracy and advantage of true convergence. The matrix model generation time is dependent on the circuit size. We have significantly reduced the number of models to be built by utilizing device matching properties of analog circuits. When matrix models are used in optimization based synthesis, partial model evaluation is done to speed up the matrix computation in successive iterations.

4 Comparison of Circuit Matrix Models and Performance Models

Performance estimation of analog circuits can use either system level models or performance level models. It is known that the relation between performance parameters such as UGF, PM and device sizes is extremely nonlinear [15, 5]. Sophisticated modeling approaches such as posynomials, neural networks are needed for modeling these severely nonlinear responses. However, these approaches too give significant errors [13]. We have observed that system matrix elements have lesser nonlinearity and can be accurately modeled.

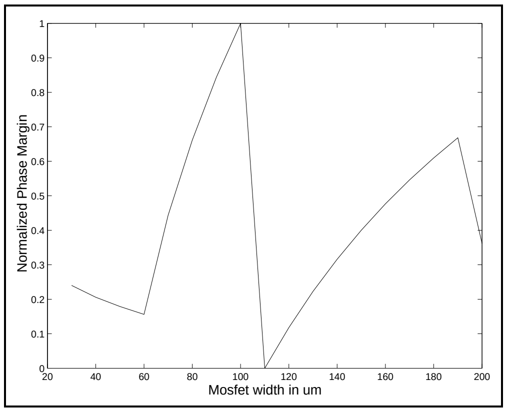

Fig. 2. Phase Margin vs. Device Width of OTA

Consider the operational Transconductance Amplifier (OTA) in fig. 4(a) as an example. We generated plots of performance parameters against device sizes and matrix elements against device sizes. Figures 2, 3 are representative plots of performance (PM) and matrix element (gds_M4). We can intuitively state from the figures that the matrix element is less nonlinear. The qualitative observation that matrix elements have less nonlinearity is now backed with two quantitative measures:

1. entropy of response curves
2. variance of local differentials

Entropy measures the complexity of a response curve [16], higher the entropy more complex the response. Entropy is calculated by definition from [17]. Variance of local first order differentials measures smoothness of a response, with lesser variance indicating greater smoothness. A response that has low entropy and is smooth is less complex to model. Worst case entropy and local variance values among all matrix elements is

Table 1. Entropy and Local Differential Variation of OTA

Response Variable	**Variance of Local Differential**	**Entropy**
Matrix Element (Worst Case)	0.0316	0.8565
Gain	0.0369	0.8072
UGF	0.0422	0.7076
Gain Margin	0.2318	1.5674
Phase Margin	0.4248	3.8017
Results are on a dataset of 2000 points		

shown in Table 1. The table also shows the entropy and local variance for performance parameters. Phase and Gain Margins have entropy and local variance an order greater than the matrix elements. From these qualitative and quantitative measures we can infer that matrix elements are less nonlinear and can be modeled with greater accuracy than their performance counterparts.

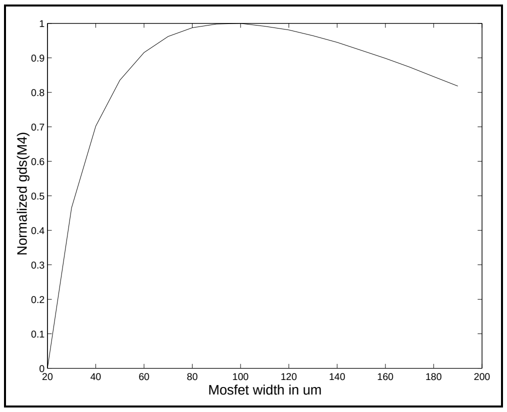

Fig. 3. Matrix Element vs. Device Width of OTA

5 Modeling Methodology

The matrix elements show a linear or curvilinear variation with respect to design variables. We model the response matrix by polynomial regression. The input variables of the model, usually the transistor widths, are normalized on a [0,1] range using eq.(2), since for polynomial regression it is important that higher order terms do not have high collinearity with lower order terms [18].

$$x_{transformed} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

The response is modeled using a least squares (LS) polynomial fit given by the following equation:

$$Y(x_1...x_n) = \beta_0 + \beta_1 x_1 + .. + \beta_n x_n + \beta_{11} x_1^2 + \beta_{12} x_1 x_2 + .. \qquad (3)$$

(where β_is are coefficients of the polynomial fit.)

It is observed that the capacitance sub-matrix terms are highly collinear with respect to the design variables, and lower order polynomials are sufficient for modeling them. The conductance sub-matrix containing terms such as gm, gds etc are more non-linear and are modeled by higher order polynomials. Once the response model within acceptable error limits is obtained by a LS fit, the regression coefficients are saved. The response at any unknown design point within the model bounds can now be predicted by simply plugging the input variable values in the model given by eq.(3). This makes response prediction extremely fast.

5.1 Circuit Matrix Generation

The first step in matrix macromodeling is generation of the circuit matrix. Subsequently values for the matrix are obtained in the design space and the matrix is modeled. Since we want to model the circuit matrix in terms of its elements, we would like to reduce the number of matrix elements to be modeled to as few as possible. To enable this reduction, we take advantage of:

– *matched* element identification
– *reverse* element identification

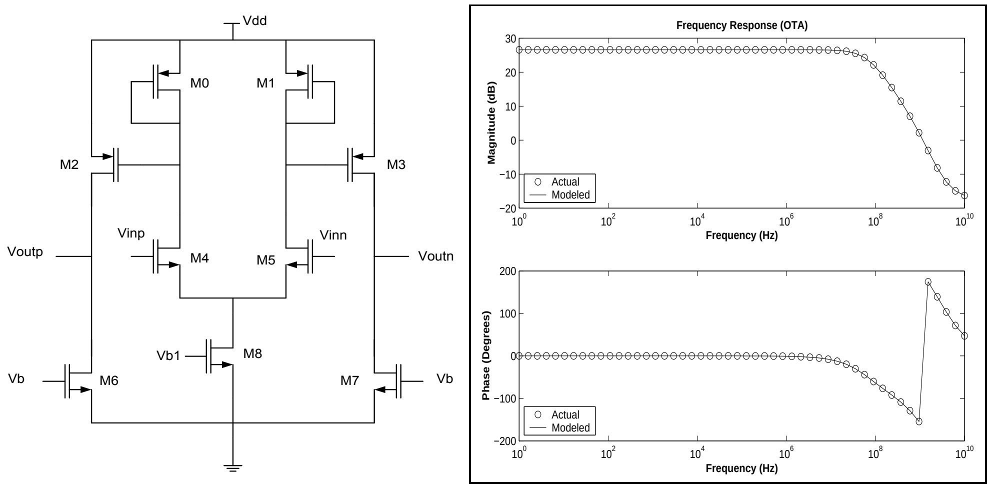

Fig. 4. (i) OTA schematic (ii) Actual vs. Modeled Frequency Response of OTA

In the OTA circuit fig 4, we can see that the transistor pairs $M0 - M1, M2 - M3, M4 - M5$ and $M6 - M7$ are matched. Using the half circuit concept [19] we know that the small signal values of the matched pairs will be equal. Thus, if the matrix elements are linear combinations of small signal values of matched elements, even these matrix elements will be identical. As a simple example, in the OTA the pairs $M0 - M1$ and $M2 - M3$ are matched and $gm0 = gm1$ and $gm2 = gm3$. If the circuit matrix has two elements, one being $gm0 + gm2$ and the other being $gm1 + gm3$, we know that these two elements will always have the same value. Thus a single model will be sufficient for both these matrix elements. With the MNA formulation we have seen that such identical elements occur at many places in the circuit matrix.

It is also observed that in the MNA matrix, some elements appear only with a reversal of polarity. For example, one matrix element is $gm4$ and the other is $-gm4$. It is possible to use a single model for elements that occur with opposite signs. Thus, we observed two properties of the circuit matrix elements which will help us reduce the number of elements to be modeled.

When the circuit matrix is generated through its MNA formulation, the matrix coefficients are first generated in a symbolic form to identify identical and reverse polarity elements. For our benchmarks, the number of non-zero coefficients in the original matrix versus the number of coefficients that need modeling after reduction is depicted in Table 2. The achievable reduction depends on the topology and the number of matched elements.

Table 2. Reduction of Matrix Elements

Benchmark	**Original Matrix Elements**	**Elements after Reduction**	**Percentage Reduction**
TSO	43	24	44
OTA	39	14	64
Differential Amplifier	145	61	58

5.2 Data Generation and Modeling

As with any modeling approach, we first need to generate raw data on which the model will be built. The data is obtained by performing a spice operating point analysis at a number of design points and storing values of circuit matrix elements. We have used random numbers drawn on a uniform distribution of the device ranges to sample the entire design space. About 2000 random data points are sampled for circuits with smaller design space such as the two stage amplifier, OTA and about 4000 points for circuits such as the differential amplifier with a larger design space . We have used high order polynomial response surface models for the circuit matrix as these give adequate accuracy.

For polynomial models it is important to choose the order appropriately since choosing a lower order than necessary will give an erroneous model, whereas choosing a higher order will cause overfitting. In our benchmark circuits we find that polynomials

Table 3. Modeling Accuracy for OTA

Matrix Element	**Polynomial Model Order**	**Max Error (%)**	**Mean Error (%)**	**Std Dev (%)**
C_{11}	2	0.0667	0.0129	0.0094
C_{13}	2	0.0439	0.0107	0.0085
C_{16}	3	0.0467	0.0122	0.0089
C_{33}	3	0.0908	0.0153	0.0155
C_{35}	1	0.0409	0.0102	0.0085
C_{55}	1	0.0430	0.0104	0.0081
C_{66}	3	0.0488	0.0115	0.0087
G_{11}	2	0.0641	0.0179	0.0119
G_{13}	6	0.2016	0.0207	0.0266
G_{15}	6	0.1879	0.0198	0.0252
G_{51}	7	0.3574	0.0602	0.0508
G_{55}	6	0.1888	0.0196	0.0254
G_{61}	4	0.1423	0.0202	0.0192
G_{66}	4	0.1256	0.0268	0.0206

with order 8 and beyond tend to overfit. We predefine the maximum order as 7 for our models. The model error is calculated using eq.(4). We define an error of 0.5% as the allowable model error.

Starting with a linear model, if the model error is less than the allowable error, that order is chosen, else we fit a polynomial with one higher order. This is done till the maximum order of 7 is reached. In some cases, increasing the order, gives very little return in terms of error reduction (the adjusted R^2 regression criterion), in which we use a lower order model to avoid complexity. Algorithm 1 shows the entire modeling procedure. Table 3 shows the modeling accuracy for each matrix element of the OTA matrix. The frequency response of the OTA with the original system matrix versus the modeled matrix at a random design point is shown in fig. 4. It is seen that the two frequency responses match extremely well.

$$ModelError = \left| \frac{ActualValue - PredictedValue}{ActualValue} \right| * 100\% \qquad (4)$$

After the model has been generated using sample data, the next step is model validation. Validation is necessary to ensure that the regression model obtained holds good for the entire design space and not just the sample data used to build the model. Validation of the model involves assessing the effectiveness of the model against an independent set of data and is essential if confidence in the model is to be expected [20]. For the purpose of validation we generate an independent set of random data points, 1000 data points for smaller circuits and 2000 points for larger circuits. The validated matrix model is used for estimating the performance of the analog circuit.

Algorithm 1 Generate Matrix_Model

```
Input: circuit.spice
Output: regression coefficients for all matrix elements
$Generate\_System\_Matrix();$
$Identify\_Unique\_Elements();$
$Generate\_Data();$
$\forall$ Unique Elements do:
$order = 1;$
$done = false;$
while (!done) do
    polyfit(response, variables, order);
    Error(order) = Calc_Model_Error(order);
    if (error(order) <= max allowed err) then
        Save_Reg_Coeffs(element);
        done = true; break;
    end if
    if ((error(order)-error(order-1)) <= 1%) then
        Save_Reg_Coeffs(element);
        done = true; break;
    end if
    if (order <= max allowed ord) then
        Increment(order,1);
    else
        Save_Reg_Coeffs(element);
        done = true;
    end if
end while
```

6 Experiments and Results

We have used three benchmark circuits: the two stage amplifier (TSO), the operational transconductance amplifier (OTA) and the high gain differential amplifier (DA) for testing the accuracy of our models. The TSO [5] is a 8 transistor circuit with five design variables (fig. 5), the OTA is a 9 transistor circuit with four variables (fig. 4)and the differential amplifier [21] is a 33 transistor circuit with five variables (fig. 6). Design space reduction was done as explained in [5] to obtain the design variables. The design variables and their ranges used for the experiments (Table 4) are selected similar to earlier published performance macromodeling work of [5] to enable a comparison of results for the two methods. The design variable ranges are such that all design points lie in a valid pocket i.e. all transistors are in saturation in the given range.

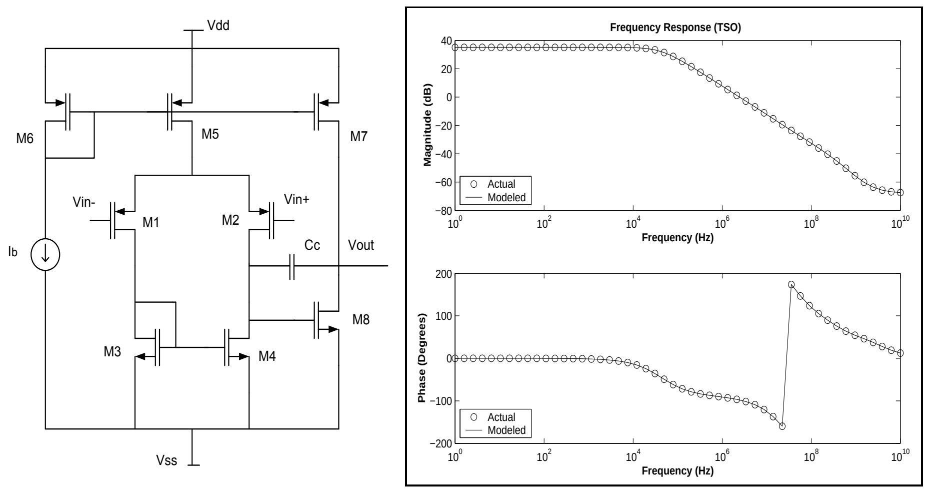

Fig. 5. (i) TSO schematic (ii) Actual vs. Modeled Frequency Response of TSO

Operating point analysis is done using Synopsys®Hspice and values for the elements of the matrix are obtained. For generating and evaluating the polynomial regression models the Matlab®Statistics Toolbox running on a 1.7GHz Pentium®M with 512 MB RAM is used.

Table 5 shows the time required to build the models and the time to estimate performance values for a given size. Table 6 shows the maximum matrix modeling errors for the benchmarks. It is seen that the elements are modeled very accurately with the maximum error about 0.5-4%. Figures 5, 7 compare the ac frequency response obtained from actual circuit matrix and the modeled circuit matrix for the TSO and Differential amplifier for a randomly chosen circuit size. The modeled response matches the actual response extremely well. We compare our model building and estimation time with a performance macromodeling approach [9] that uses support vector machines. As performance is directly modeled in the second case the estimation time is lower, but the maximum error is 10.1%.

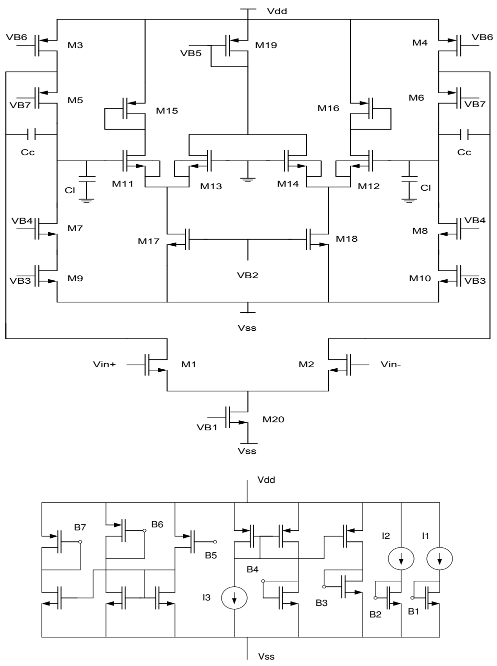

Fig. 6. DA schematic [21]

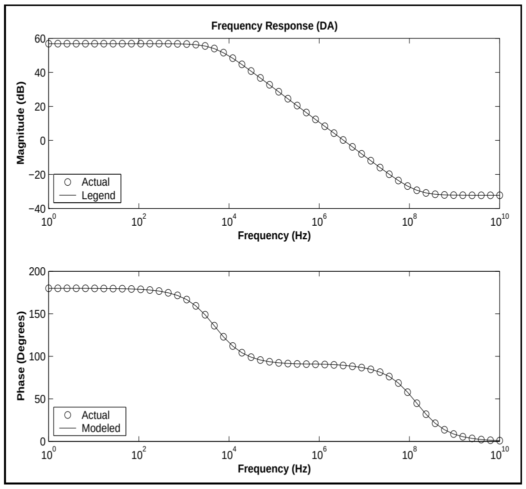

Fig. 7. Actual vs. Modeled Frequency Response of Differential Amplifier

Table 4. Design variable ranges for Benchmarks

Benchmark	**Mosfet Count**	**Number of Variables**	**Ranges**
TSO	8	5	M1-M5,M7:20-80um, Cc:2-10pF, l:2um
OTA	9	4	M2-M5:20-200um M0,M1,M6,M7:20-35um, l:2um
DA	33	5	M1-M10, 4*M25, 4*M26, 2*M23, 2*M27, 2*M28, 2*M32: 40-200um, Cc: 10-50pF, l: 4um

Table 5. Modeling and Estimation Time

Benchmark	**Modeling Time**	**Performance (All) Estimation Time**
	Matrix Modeling Approach	
TSO	$3.7min$	$0.033sec$
OTA	$16sec$	$0.021sec$
DA	$31.7min$	$0.104sec$
	Competing Approach [9]	
TSO	$131.15min$	$0.01sec$
OTA	$50.085min$	$0.001sec$

The performance parameters are calculated from the generated matrix models and results are compared with a spice simulation. Table 7 shows the maximum, mean and standard deviation of the performance estimation error for all benchmarks. The maximum error with matrix models is about 3% and the highest mean error is about 0.1%. To enable a comparison with performance macromodeling, polynomial regression models were built on the performance parameters directly. Table 8 comprises the results of directly modeling the performance. As would be expected, the errors are higher. The TSO and Differential Amplifier circuits are identical to the work of [5] which uses neural networks for performance estimation. The maximum performance estimation error in [5] is 45% and highest mean error is about 5%.

As the circuit matrix is modeled, the time for an operating point analysis is saved which can be upto 70% of the total analysis time [3]. Although the performance is not

Table 6. Worst Case Validation Error

Benchmark	**Validation Dataset Size**	**Worst case Error (%)**
TSO	1000	1.8
OTA	1000	0.51
DA	2000	4.37

Table 7. Performance Estimation Accuracy with Proposed Approach

Benchmark	Max Error (%)	Mean Error (%)	Std Dev
Two Stage Op-Amp			
Gain	0.3231	0.0301	0.0370
UGF	0.6234	0.0544	0.0623
GM	1.2944	0.0900	0.1220
PM	0.7848	0.0521	0.0833
CMRR	0.7372	0.0668	0.0818
Operational Transconductance Amplifier			
Gain	0.1555	0.0134	0.0168
UGF	0.3111	0.0232	0.0320
GM	0.3199	0.0363	0.0367
PM	1.2864	0.0597	0.1068
Differential Amplifier			
Gain	3.2670	0.1214	0.1490
UGF	2.2863	0.1473	0.1820
PM	0.7970	0.0648	0.0666

available directly and needs an extra step for its computation, the performance calculation time is much smaller than a spice evaluation. The added advantage with our method is that since the entire ac behavior is modeled, any related performance can be evaluated. Thus, if a performance parameter is required, it simply needs to be evaluated from the matrix model and a new model need not be generated for that parameter.

7 Synthesis Using Circuit Matrix Models

This section describes circuit sizing using the developed circuit matrix models. An optimization algorithm such as Simulated Annealing (SA) used for sizing works by perturbing the current solution to propose a new solution. With incremental perturbation a single parameter of the current solution is varied in every iteration. An important observation is that a design parameter affects only some elements of the circuit matrix. Thus during an SA move only the affected matrix elements are re-evaluated.

Synthesis is explained using the Differential Amplifier as an example. The Differential Amplifier has 61 matrix elements and 5 design variables. The design variables are four mosfet widths $(w_1 - w_4)$ and capacitance C_c. The correlation coefficient between matrix elements and design variables is calculated. If the p value of the correlation is less than 0.1, the correlation is considered significant. Based on the p values it is seen that w_1 affects 8 matrix elements, w_2 affects 30, w_3 and w_4 affect 30 and 36 elements respectively whereas 11 elements are dependent on C_c. Thus, with a maximum of 36 elements are evaluated when w_4 changes and only 8 elements need to be evaluated if w_1 changes.

During synthesis, new solutions are proposed by incrementally updating the current solution. Based on the design variable that is perturbed, the affected matrix elements are calculated from their models. The circuit matrix is then solved for various values

Table 8. Estimation Accuracy by Direct Performance Modeling

Benchmark	**Polynomial Order**	**Max Error (%)**	**Mean Error (%)**	**Std Dev**
Two Stage Op-Amp				
Gain	4	0.2523	0.0285	0.0272
UGF	7	14.06	1.4307	1.2449
GM	5	3.4894	0.6363	0.4933
PM	6	2.6748	0.3855	0.3463
CMRR	5	0.4704	0.0512	0.0555
Operational Transconductance Amplifier				
Gain	5	0.7582	0.0784	0.0878
UGF	6	8.4583	1.3731	1.0731
GM	7	7.1955	0.6016	0.7836
PM	7	$3.3e3$	186.31	411.50
Differential Amplifier				
Gain	4	6.2527	1.2365	1.1651
UGF	2	35.8732	9.9997	9.3995
PM	4	0.5234	0.0490	0.0567

of the frequency variable 's'. This gives the frequency response for the circuit. Values such as gain, bandwidth are calculated from the frequency response and compared with the given specification. The sizing algorithm terminates when a solution satisfying the required specifications is found or if no solution can be found in reasonable amount of time. Circuit sizing results for the Differential Amplifier by Simulated Annealing are given in Table 9. The target specifications are given in column 1. Column 2 gives the predicted values for performance at the sizing solution, and column 3 is the actual spice verified values for the sizing solution. It can be seen that the predicted and actual performance values match very well. Thus circuit matrix models are very accurate and can be used for performance prediction during synthesis.

Table 9. Differential Amplifier Synthesis with Partial Model Evaluation

Performance Specification	**Estimated**	**Actual**
Gain $\geq$ 78 dB	78.86	79.04
UGF $\geq$ 25 MHz	25.38	24.95
Phase Margin $\geq$ 88 Deg	88.43	88.72

8 Techniques for Faster Synthesis

This section describes two techniques to speed up the synthesis of analog circuits using circuit matrix models. Both techniques are based on reducing the time required to evaluate matrix elements from their models during each iteration of the synthesis run.

8.1 Speedup by Hashing

Here we store computed matrix elements in hash tables which are fetched when required to reduce model evaluation time. The SA algorithm used for synthesis starts with an initial random sizing solution and continuously makes incremental changes to the solution till the target specifications are satisfied. Although exactly same solutions are rarely encountered during the synthesis run, sub-solutions often get repeated. For example, for a circuit with 4 design variables two solutions proposed at different SA iterations are v_1 = 10, v_2 = 20, v_3 = 30, v_4 = 40 and v_1 = 10, v_2 = 40, v_3 = 60, v_4 = 40. Although, the solutions proposed are different, the sub-solution v_1 = 10, v_4 = 40 is the same in both cases. Thus, if we save matrix elements dependent on v_1, v_4 computed at the earlier iteration, they can be simply fetched and model evaluation time is saved.

Each matrix element is a function of a subset of design variables. We group all matrix elements that depend on the same design variable subset into a class called *hash class*. One hash table is constructed for each hash class. The sub-solution and the corresponding matrix element values for each hash class are stored in the hash table. During synthesis, the hash table is queried to check if the sub-solution was encountered previously. If the sub-solution was visited earlier, all the hash class elements can be obtained *at once* from the hash table, otherwise they are evaluated from their models and stored.

Thus, the steps involved in using hash tables for faster model evaluation are:

– group matrix elements into hash classes
– initiate one hash table for each hash class
– retrieve matrix elements from the hash table when possible

For example, the 61 matrix elements of Differential Amplifier circuit are divided into 9 different hash classes. The largest hash class has 16 matrix elements while the smallest has 1 element. Hash tables are constructed using a R-B tree data structure [22]. Both insertion and querying is performed in O(log n) time. Using hash tables to avoid matrix element recomputation gives an average synthesis speedup of 2.8x measured over a set of 35 synthesis experiments of the DA. Details of the procedure of using hashing for expedited synthesis and further experimental results can be found in [23].

8.2 Speedup by Near Neighbor Searches

Hash tables store and reuse matrix element values, thus reducing the time required to evaluate the matrix from its element models. Hashing is useful only when a newly proposed sub-solution *exactly* matches a previously visited one. However, during a synthesis run there may be many sub-solutions close to previously visited solutions without matching exactly. In such cases, computing values of matrix elements incrementally from a close (neighboring) previously visited design point helps in saving matrix model evaluation time.

For example, consider a matrix element M dependent on variables v_1, v_2, v_3, therefore M = f(v_1, v_2, v_3). A first order Taylor series expansion for M is given by:

$$dM = \frac{\partial M}{\partial v_1} \cdot dv_1 + \frac{\partial M}{\partial v_2} \cdot dv_2 + \frac{\partial M}{\partial v_3} \cdot dv_3 \tag{5}$$

If the value of M is already calculated at some design point (v_1 = x, v_2 = y, v_3 = z), its value can be quickly obtained at the neighboring design point (v_1 = x + Δx, v_2 = y + Δy, v_3 = z + Δz) using eq.(5). Thus instead of evaluating a higher order polynomial model, the element evaluation is done using the linear equation above making the computation faster. For all design points visited during synthesis, the value of the computer matrix element M and its associated differentials ∂M/∂v are stored and reused to compute matrix value at many neighboring points. Thus, computing matrix element values using the above method require the following steps:

– find a previously evaluated neighbor of the currently proposed sub-solution
– obtain the matrix element value and the value of differentials at the neighboring point
– calculate the matrix element value at the new point using eq.(5)

A good neighbor searching algorithm is essential for the success of this method. An optimal (near) neighbor search algorithm proposed by Arya *et al.* [24] can be successfully applied for this purpose. Since only few of the matrix elements require actual evaluation from their models the speedup by this method is much more than hashing alone. For the Differential Amplifier synthesis, computing matrix elements incrementally from its neighbors results in a speedup of about 13x over simple matrix element evaluation whereas with hashing it was only 2.8x. Further details of this method and experimental results can be found in [25].

9 Synthesizing circuit with different specifications

In the case of performance macromodels, performance data is gathered for certain parameters. Models are developed for these parameters and are used for synthesis. Using models makes performance evaluation faster than simulation and expedites the synthesis process. However, only the parameters for which performance models have been developed can be included in the synthesis. On the other hand, with circuit matrix models the target specifications need not be known beforehand. Performance parameter values required are calculated from the frequency response obtained from solving the circuit matrix. This is demonstrated with a band pass filter circuit.

9.1 Additional specifications for synthesis

Fig. 8 shows a 2nd order band pass filter with a Sallen Key implementation. Design variables identified for the filter synthesis are widths of mosfets M9, M7, resistor R3, capacitors C1, C2. The filter is to be synthesized with target specifications for gain, bandwidth and center frequency. The synthesis engine proposes sizes for design variables. For each set of sizes, the circuit matrix is obtained from evaluating the element models. Substituting the ’s’ variable with frequency values gives the frequency response of the circuit. The values of gain, bandwidth and center frequency are calculated from the frequency response till a sizing solution meeting the specifications is obtained. The synthesis results are shown in table 10.

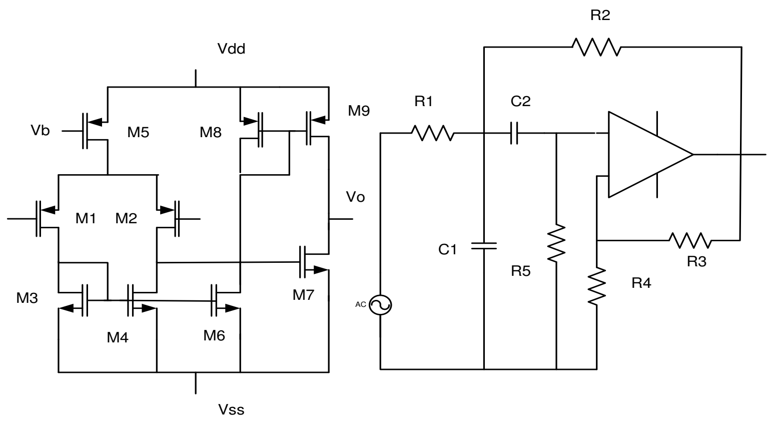

Fig. 8. (i) Amplifier used in bpf (ii) Active band pass filter schematic

Suppose, the filter has to be synthesized for another application where target specification include FP1 (frequency at the edge of the start of the pass band), FP2 (frequency at the edge of the end of the pass band) in addition to gain, bandwidth and center frequency. With circuit matrix models, synthesizing circuit with these additional specifications is simple. The cost function is changed to include the additional specifications. Both FP1 and FP2 are calculated from the frequency response along with the other specifications and synthesis procedure is the same as before. Table 10 shows the synthesis results.

Table 10. Synthesis results for the band pass filter

Performance Specification	**Estimated**	**Actual**
Gain ≥ 14 dB	15.18	15.44
Bandwidth ≥ 2000 Hz	2409	2417
Center Frequency ≥ 2500 Hz	2590	2592
FP1 ≥ 200 Hz	273	273
FP2 ≤ 15000 Hz	12109	12103

9.2 Alternate forms of target specifications

With circuit matrix models, circuits can be synthesized with alternate forms of target specifications and not necessarily performance parameters alone. In the next experiment, we synthesize the filter circuit with the specifications given in the form of a frequency response instead of parameters such as gain, bandwidth. The input specifications are in terms of the magnitude response at different frequencies (phase response specifications can be added similarly). The cost function is changed to include frequency response parameters instead of performance parameters. The rest of the synthesis process remains the same. Figure 9 shows the synthesis results. The blue line shows the specified response. The red dots show the frequency response achieved by the target circuit and the green dots are the SPICE frequency response for the sized circuit. Thus synthesis is possible with alternate specifications only by changing the cost function.

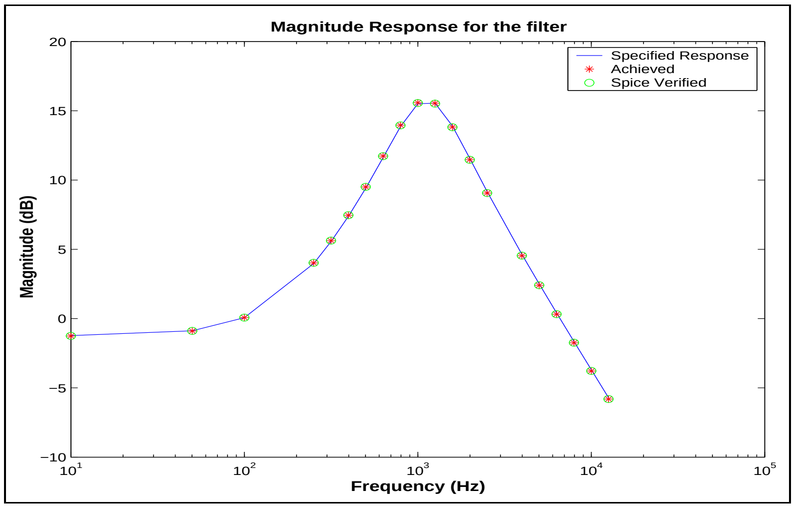

Fig. 9. Synthesizing filter from Frequency Response

10 Conclusion

Two methods for performance estimation of analog circuits, performance modeling and circuit modeling, are compared. It is demonstrated that the circuit matrix can be accurately modeled using polynomial regression. The number of coefficients that need to be modeled are significantly reduced by taking advantage of transistor matching. The accuracy of the proposed method is validated through experiments on three operational amplifier benchmarks. Techniques such as hashing and near neighbor searches can significantly speed up the synthesis process. Using circuit matrix models, synthesis can be performed for different types of specification such as performance parameters or frequency response.

References

1. M. Krasnicki, R. Phelps, R. A. Rutenbar, and L. R. Carley, “MAELSTROM: Efficient simulation-based synthesis for custom analog cells,” in *Proc. - 36th Design Automation Conference*, pp. 945–950, 1999.

2. M. J. Krasnicki, R. Phelps, J. R. Hellums, M. McClung, R. A. Rutenbar, and L. R. Carley, “ASF: a practical simulation-based methodology for the synthesis of custom analog circuits,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 350–357, 2001.
3. M. Ranjan, W.Verhaegen, A.Agarwal, H.Sampath, R.Vemuri, and G.Gielen, “Fast, layout-inclusive analog circuit synthesis using pre-compiled parasitic-aware symbolic performance models,” in *Proc. DATE*, p. 10604, 2004.
4. T. McConaghy and G. Gielen, “Double-strength CAFFEINE: Fast template-free symbolic modeling of analog circuits via implicit canonical form functions and explicit introns,” in *Proceedings of the conference on Design, Automation and Test in Europe*, pp. 269–274, 2006.
5. G. Wolfe and R. Vemuri, “Extraction and use of neural network models in automated synthesis of operational amplifiers,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* **22**, pp. 198–212, Feb. 2003.
6. S. Doboli, G. Gothoskar, and A. Doboli, “Extraction of piecewise-linear analog circuit models from trained neural networks using hidden neuron clustering,” in *Proceedings of the conference on Design, Automation and Test in Europe*, p. 11098, 2003.
7. T. Kiely and G. Gielen, “Performance modeling of analog integrated circuits using least-squares support vector machines,” in *Proceedings of the conf. on Design, Automation and Test in Europe*, p. 10448, 2004.
8. F. D. Bernardinis, M. I. Jordan, and A. S. Vincentelli, “Support vector machines for analog circuit performance representation,” in *Proc. DAC ’03*, pp. 964–969, 2003.
9. M. Ding and R.Vemuri, “A combined feasibility and performance macromodel for analog circuits,” in *Proc. 42nd Design Automation Conference*, pp. 63–68, 2005.
10. D. Han and A. Chatterjee, “Adaptive response surface modeling-based method for analog circuit sizing,” in *Proceedings. IEEE International SOC Conference*, pp. 109–112, 2004.
11. G. Wolfe and R. Vemuri, “Adaptive sampling and modeling of analog circuit performance parameters with pseudo-cubic splines,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 931–938, 2004.
12. H. Liu, A. Singhee, R. A. Rutenbar, and L. R. Carley, “Remembrance of circuits past: macromodeling by data mining in large analog design spaces,” in *Proc. DAC ’02*, pp. 437–442, 2002.
13. T. McConaghy and G. Gielen, “Analysis of simulation-driven numerical performance modeling techniques for application to analog circuit optimization,” in *Proc. - ISCAS*, pp. 1298–1301, May 2005.
14. R. Rutenbar, G. Gielen, and J. Roychowdhury, “Hierarchical modeling, optimization, and synthesis for system-level analog and RF designs,” *Proceedings of the IEEE* **95**(3), pp. 640–669, March 2007.
15. P. Mandal and V. Visvanathan, “Macromodeling of the A.C. characteristics of CMOS op-amps,” in *Proc. International Conference on Computer-Aided Design.*, **7**, pp. 334–340, Nov. 1993.
16. France, Michel Mendes and Henaut, Alain, “Art, therefore entropy,” *Leonardo* **27**(3), pp. 219–221, 1994.
17. A. Denis and F.Cremoux, “Using the entropy of curves to segment a time or spatial series,” in *Mathematical Geology*, **33**, pp. 899–914, Nov. 2002.
18. M. Kutner, C. Nachtsheim, W. Wasserman, and J. Neter, *Applied Linear Regression Models*, McGraw-Hill/Irwin, 2003.
19. B. Razavi, *Design of Analog CMOS Integrated Circuits*, McGraw-Hill, Inc., 2000.
20. J. Rawlings, S. Panstula, and D. Dickey, *Applied Regression Analysis : A Research Tool*, Springer, 2001.

21. J. M. Cohn, *Analog Device-Level Layout Automation*, Kluwer Academic Publishers, Norwell, MA, USA, 2000.
22. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, 2001.
23. A. Pradhan and R. Vemuri, “On the use of hash tables for efficient analog circuit synthesis,” in *Proceedings of the International Conference on VLSI Design*, Jan 2008.
24. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching in fixed dimensions,” *Journal of the ACM* **45**(6), pp. 891–923, 1998.
25. A. Pradhan and R. Vemuri, “Fast analog circuit synthesis using sensitivity based near neighbor searches,” in *Proceedings of the conference on Design Automation and Test in Europe (To Appear)*, Mar 2008.