A Methodology for Statistical Behavioral Fault Modeling

Zheng Rong Yang and Mark Zwolinski
Department of Electronics and Computer Science
University of Southampton, SO17 1BJ, UK
Email mz@ecs.soton.ac.uk

Abstract

This paper presents a novel algorithm for statistical behavioral modeling, based on two statistical techniques: mutual information and Bootstrap. In contrast to Euclidean distance calculations, clustering faults by measuring the mutual information (entropy) between two fault populations is more efficient and robust. Employing the bootstrap technique results in a significant reduction of expensive Monte Carlo simulation time.

Keywords: behavioral modeling, bootstrap, mutual information, Monte Carlo.

1. Introduction

Simulating a complicated circuit with macromodels or behavioral models is a widely employed methodology for electronic circuit design because it saves time and simplifies the design process [1-5].

A macromodel usually denotes a simplified form of a sub-circuit, which may be repeated many times in a design. There are two ways to realize a macromodel: as a simplified circuit or as a statistical macro.

A behavioral model describes a sub-circuit with explicit equations. Even when circuit behavior, in particular, that of a faulty circuit, is not easy to derive, behavioral modeling still plays an important part in device level or block design, such as FETs or opamps. There are also two ways to realize behavioral models, analytical and statistical. An analytical description of a sub-circuit is usually generated by a designer. It should be noted that even using an analytical approach, behavioral modeling still needs verification which can be expensive.

A statistical behavioral model can be built in the same way as a statistical macromodel. The difference is that a statistical macromodel does not include an explicit transfer function. Furthermore, simulation using statistical macromodels employs table look-up techniques, while simulation using statistical behavioral models is based on a few mathematical calculations, and may therefore have a cheaper computational cost.

One application of behavioral modeling is in analogue fault simulation. Exhaustive fault modeling is very expensive. The common method is to cluster faults into several groups by hand. Fault simulation only needs to be done for each group rather than for each fault. The criterion for fault clustering is to measure the Euclidean distance between fault responses [4]. This simple measurement is not robust when the fault populations are not distributed in a homoscedastic (normal distributions with the same variances) manner. In this paper, fault clustering is performed by applying Shannon's theorem, or mutual information theory [10-12]. Because the populations formed by the faulty behavior of circuits are distributed in a heteroscedastic (normal distributions having different variances) manner [15], mutual information exploits this characteristic during fault clustering. However, mutual information is based on a population probability density calculation. To calculate this explicitly would require a large computational cost in the form of Monte Carlo simulations.

Hence, the Bootstrap method [7] is applied to reduce the number of Monte Carlo

simulations. The Bootstrap method is a well-established robust statistical methodology. Although Bootstrap has been widely employed in the social sciences [8] and signal processing [9], there are few reports of applying Bootstrap to electronics design [13].

2. Mutual information theory

A simple explanation of mutual information theory is given here. Suppose there are two persons, A and B. According to mutual information theory, they will not be best friends unless A is the best friend of B and B is the best friend of A.

Mutual information theory aims to minimize the entropy within a system. As described in [14], mutual information is measured by the difference between the initial uncertainty

$$H(\Psi) = -\sum_{g\in\Psi} p(g)\log p(g)$$

and the average uncertainty

$$H(\Omega) = -\sum_{\mathbf{x}\in\Omega} p(\mathbf{x}) \sum_{g\in\Psi} p(g\,|\,\mathbf{x})\log p(g\,|\,\mathbf{x})$$

This leads to the system uncertainty

$$I(\Psi,\Omega) = \sum_{g\in\Psi,\mathbf{x}\in\Omega} p(g,\mathbf{x})\log\frac{p(g,\mathbf{x})}{p(g)p(\mathbf{x})}$$

In the above equations, $\Psi\in\Re^1$ is a space containing classes, $\Omega\in\Re^d$ is a space containing features (d is the number of dimensions), $p(g, \mathbf{x})$ is the joint probability for the gth class and the feature $\mathbf{x}$, $p(g)$ is the apriori probability for the gth class and $p(\mathbf{x})$ is the (known) probability of the feature $\mathbf{x}$.

The work in [14] aimed to extract the features $\Omega=\{\mathbf{x}\}$ for a given number of classes $\Psi=\{g\}$. An optimal set of features is selected based on minimizing the system uncertainty.

Here, however, the number of classes is unknown. We therefore define the initial uncertainty, average uncertainty and system uncertainty as above, but now $p(\mathbf{x})$ is the *unknown* average pattern probability for the pattern $\mathbf{x}$.

It is not necessary that all the populations in a data space (here, one population refers to one group of faults) are distributed homoscedastically. If the Euclidean distance is used for fault clustering, it is difficult to define a suitable threshold for all the fault populations because some fault populations have a large deviation and some fault populations have a small deviation [15]. A low threshold leads to some faults, which should be grouped together, not being grouped while a large threshold will result in some faults being grouped, that should not be.

Hence, fault clustering using mutual information between different fault populations should perform better than using a Euclidean distance calculation.

3. Probability density estimate

The average class probability is calculated from

$$p(g) = N_g / N$$

where, N is the number of total patterns and N_g is the number of patterns, that belong to the gth class. The average pattern probability is

$$p(\mathbf{x}) = N_{\mathbf{x}} / \left(\pi\sigma^d\right)$$

where σ is defined to be

$$\sigma = \max\left\{\left\|\mathbf{x}_i - \mathbf{x}_j\right\| \,|\, \mathbf{x}_i\in\Omega \,\&\, \mathbf{x}_j\in\Omega\right\}$$

and $N_{\mathbf{x}}$ is the number of patterns that fall in the area centered at $\mathbf{x}$ with the radius σ. The strategy for selecting σ is to ensure that all the probabilities are not be zero and hence we will not meet numerical problems when calculating the entropy. The joint probability is calculated from

$$p(g,\mathbf{x}) = N_{\mathbf{x}}^{g} / \left(\pi\sigma^d\right)$$

where $N_{\mathbf{x}}^{g}$ is the number of patterns that fall in the area centered on $\mathbf{x}$ with the radius σ belonging to the gth class.

By the definition of the average probability density p(**x**) and the joint probability density p(g,**x**), above, it is not difficult to derive the conditional probability

$$p(g \mid \mathbf{x}) = \frac{p(g,\mathbf{x})}{p(g)} = \frac{N_{\mathbf{x}}^{g}}{N_{\mathbf{x}}}.$$

4. Bootstrap method

In section 2, it was argued that using mutual information to cluster the fault populations will make the behavioral modeling procedure more efficient and robust. The most important thing is that applying mutual information theory can make behavioral modeling automatic. Mutual information is calculated based on the average probability and joint probability of populations. For probability calculation, we prefer to have an information-rich data space. However, the generation of such information in this case is impractical because the Monte Carlo simulation of a large number of faults subject to process variations is very expensive. The principle of reducing the number of Monte Carlo simulations by applying the Bootstrap technique has been addressed in [13].

The basic principle of Bootstrap is the random replication of the original sample. We wish to obtain a statistical estimate, *S*, from a sample **x**, where,

$$\mathbf{x} = \{x_1, x_2, \cdots, x_i, \cdots, x_n\}.$$

The first step of the Bootstrap method is to generate a bootstrap sample,

$$\mathbf{x}^{B} = \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^b, \cdots, \mathbf{x}^m\}$$

where,

$$\mathbf{x}^{b} = \{x_1^b, x_2^b, \cdots, x_n^b\}.$$

$\mathbf{x}^{b}$ has the same dimensionality as **x** and the elements of $\mathbf{x}^{b}$ are randomly drawn from the original sample. For example, if we have a set {1.0, 3.1, 9.2, 6.7}, one of the bootstrap samples might be {1.0, 3.1, 3.1, 6.7}. From $\mathbf{x}^{B}$, a series of new statistical estimates can be calculated as

$$\mathbf{S}^{B} = \{S^1, S^2, \cdots, S^m\}.$$

A robust estimate of *S* is then

$$S^{*} = E\left(S^{b}\right).$$

This is particularly effective when the original sample is expensive or difficult to obtaining. Bootstrap has two good asymptotic properties for a statistical estimate [7]. The first is the asymptotic property of the estimate. When the bootstrap draw number becomes large, the confidence interval will tend to converge. The second is the asymptotic property of the distribution. Even using small runs of a Monte Carlo simulation, Bootstrap draws asymptote to a normal distribution [13].

5. Parameter estimation

Numerical approximation is commonly used to estimate the parameters of a regression function. Steepest descent is one such method

$$\Delta w = -\eta \nabla$$

where η is a small coefficient, ∇ is the first-order derivative and *w* is the parameter to be estimated. However, this method can oscillate and a damping factor, α, is added to control the oscillation

$$\Delta w^{t+1} = -\eta \nabla^{t+1} + \alpha \Delta w^{t}.$$

6. The behavioral modeling algorithm

The algorithm proposed here is as follows.

Step 1. Prepare a circuit for simulation and insert all the possible or required faults into the circuit, one at a time. Label these faulty circuits together with the fault-free circuit Ω^0.

Step 2. Define a regression function for all the faults and fault-free circuit $\Im(\mathbf{x},\mathbf{w})$, where $\mathbf{x}\in\Omega^0$ and $\mathbf{w}\in\Re^h$ (h is the dimension of the parameter space).

Step 3. Run M Monte Carlo simulations for all the faulty circuits as well as the fault-free circuit

$$\Omega^0 \xrightarrow{Monte\,Carlo} \Omega^M$$

Step 4. Extract N bootstrap samples from Ω^M

$$\Omega^M \xrightarrow{Bootstrap} \Omega^B$$

thus Ω^B is the final data ready for modeling.

Step 5. Cluster the patterns in Ω^B into k groups so that

$$\Omega^B = \cup\Omega_k^B.$$

Step 6. Run a parameter estimation for a regression function $\hat{\Im}(\mathbf{x},\mathbf{w})$ on Ω^B. The objective of the parameter estimation is to minimize the error between $\Im(\mathbf{x},\mathbf{w})$ and $\hat{\Im}(\mathbf{x},\mathbf{w})$.

7. Experimental results

Figure 1 shows a differential amplifier composed of nine MOS transistors.

We inserted 24 possible short faults into this circuit. Table 1 lists the faults, the number in brackets indicates the transistor number and other two digits denote the terminals of the transistor1, where a short fault is realized by a resistor with a resistance of 10 ohms. Ten Monte Carlo DC sweep simulations were conducted for each faulty circuit as well as for the fault-free circuit. Including the fault-free circuit, we have 25 patterns in Ω^0 and 250 patterns in Ω^M. After clustering using mutual information theory, we obtained eight groups, see Table 1. Thus 25 faults (including the fault-free) were reduced to eight classes. This means that we only need to build a family of eight groups of parameters for the regression function

$$\{\Im_1(\mathbf{x}\in\Omega_1,\mathbf{w}_1),\cdots,\Im_8(\mathbf{x}\in\Omega_8,\mathbf{w}_8)\}.$$

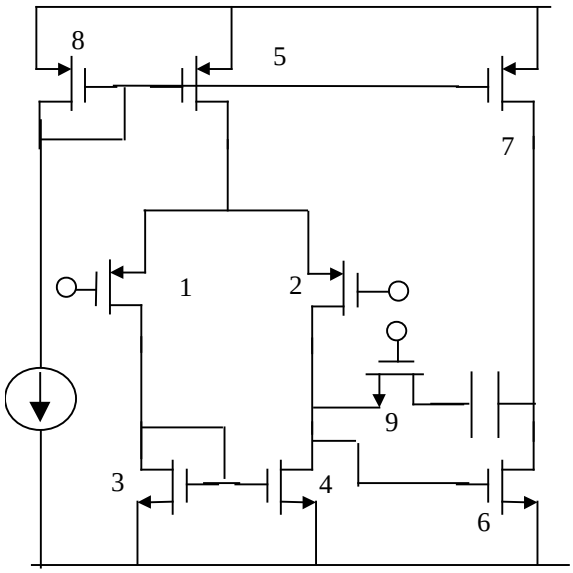

Figure 1 Example circuit

Table 1 Fault simulation results

No.	Faults	class	No.	Faults	class
0		1	13	(5,g,d)	1
1	(1,s,g)	2	14	(6,s,g)	6
2	(1,s,d)	2	15	(6,s,d)	3
3	(1,g,d)	2	16	(6,g,d)	2
4	(2,s,g)	2	17	(7,s,g)	5
5	(2,s,d)	3	18	(7,s,d)	7
6	(2,g,d)	3	19	(7,g,d)	8
7	(3,g,s)	3	20	(8,s,g)	5
8	(4,d,g)	4	21	(9,d,g)	3
9	(4,d,s)	2	22	(9,d,s)	3
10	(4,g,s)	3	23	(9,g,s)	1
11	(5,s,g)	5	24	(c)	6
12	(5,s,d)	3			

We then took 100 bootstrap samples from Ω^M, giving 250000 patterns in Ω^B.

For this example, the regression function was chosen as

$$\Im = w_0\left(1-\frac{w_1}{1+e^{-w2(Vin-w_3)}}\right).$$

Hence, there were 32 parameters to estimate in total for the eight clusters. This meant that we only needed one parameter estimation routine. Of these eight regression functions, only five actually needed parameter estimation because the output for three classes was stuck at 0 V, -5 V and +5 V.

The five regression functions, which have non-stuck responses, were calculated to be

$$\Im_1 = 5.067*\left(1-\frac{1.87}{1+e^{-103.76*(Vin-0.496)}}\right),$$

$$\Im_4 = 4.561*\left(1-\frac{1.997}{1+e^{-97.161*(Vin-0.507)}}\right),$$

$$\Im_6 = -3.609*\left(1-\frac{0.0559}{1+e^{-2.779*(Vin-0.944)}}\right),$$

$$\Im_7 = 5.007*\left(1-\frac{0.013}{1+e^{-77.465*(Vin-0.509)}}\right),$$

$$\Im_8 = 3.745*\left(1-\frac{1.613}{1+e^{-81.584*(Vin-0.502)}}\right).$$

The accuracies of these behavioral models are given in Table 2 and Figures 2 to 6.

Table 2 Behavioral model accuracy

Class	Accuracy
1	0.099471
4	0.060641
6	0.046106
7	0.004699
8	0.068811

In Figures 2 to 6, the solid lines show the responses of the behavioral models and the dotted lines show the original circuit responses.

Figure 2 shows the response for group 1 (fault-free, the 13th fault and the 23th fault). It can be seen that the behavioral model and the real circuit responses are very close.

Figure 3 shows the responses for group 4. Again, the error between the real circuit output and the behavioral model is very small.

Figure 4 shows the responses for group 6. In the middle of the plot, there is a relatively large difference between the circuit responses and behavior model. This is because the transfer function for this group is different from the regression function defined above. However, the absolute difference is only about 0.1 V and the general trend of the curve is correct.

Figure 5 shows the responses for the 7th group. The circuit model response has a small slope above 0.5 V, again the regression function does not reflect this slope. However, this error is still very small because the slope is very small.

Finally, Figure 6 plots the responses of the 8th group. It can be seen that the match between the circuit response and the behavioral model is once again good.

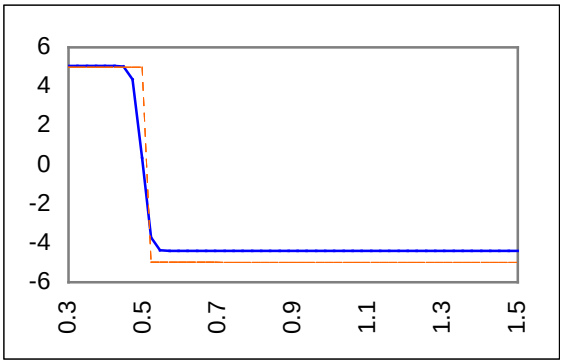

Figure 2 Circuit and behavioral model responses for group 1.

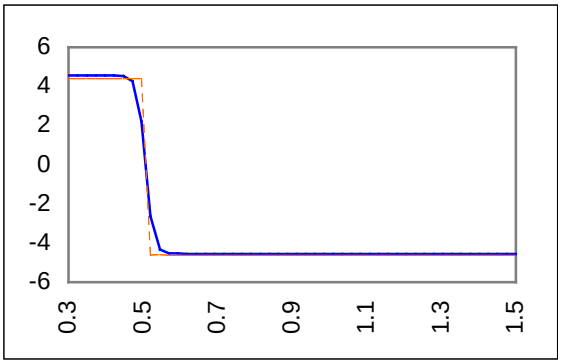

Figure 3 Circuit and behavioral model responses for group 4.

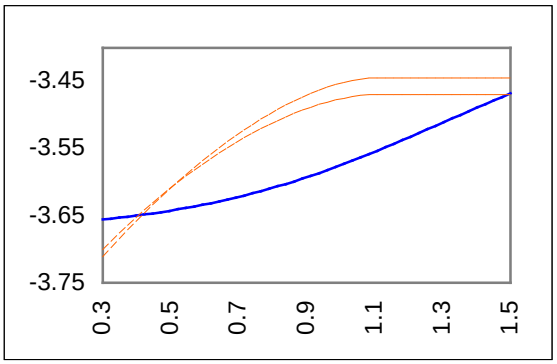

Figure 4 Circuit and behavioral model responses for group 6.

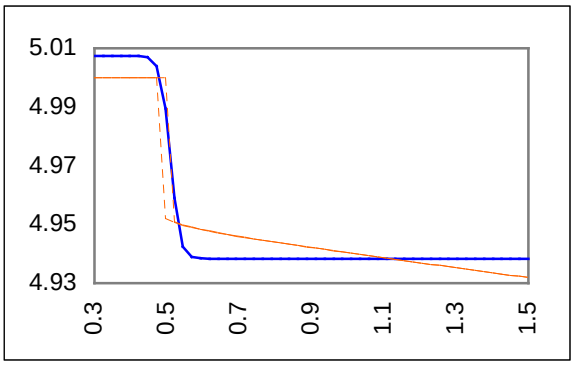

Figure 5 Circuit and behavioral model responses for group 7.

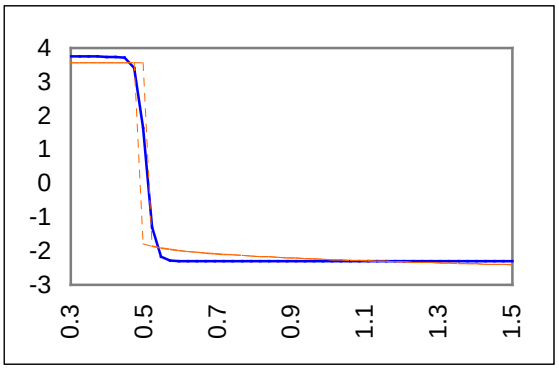

Figure 6 Circuit and behavioral model responses for group 8.

8. Conclusions

A novel algorithm for statistical behavioral fault modeling for analogue circuits has been presented in this paper. With this algorithm, exhaustive fault modeling can be avoided and manual fault clustering can be automated. The clustering mechanism is robust and efficient by using mutual information theory. Employing the bootstrap technique also reduces the Monte Carlo simulation time. Finally, the experimental results show good performance of the behavioral models.

Acknowledgements

This work has been supported by EPSRC grant GR/L35829.

References

1. Pan, C. Y. and Cheng, K. T. (1997). "Fault macro-modeling for analogue/mixed-signal circuits", *International Test Conference*, 913-922.
2. Spalding, G. R., and VanPeteghem, P. M (1990). "Design for Testability Using Behavior Models", *IEEE Trans .on Instrumentation and Measurement*, 39, no. 6, 881-885.
3. Voorakaranam, R.,Chakrabarti, S., Hou, J., Gomes, A., Cherubal, S. and Chatterjee, A. (1997). "Hierarchical specification-driven analog fault modeling for efficient fault simulation and diagnosis", *International Test Conference*, 903-912.
4. Zwolinski, M., Chalk, C. and Wilkins, B. R. (1996). "Analogue fault modeling and simulation for supply current monitoring", *European Design Automation Conference*, Paris, France, 547-552.
5. Carooll, J., Whelam, K., Prichett, S. and Bridges, D. R. (1996). "FET statistical modeling using parameter orthogonalisation", *IEEE Trans. On Microwave Theory and Techniques*, 44, no. 1, 47-55.
6. Chao, C., Lin, H. and Milor, L. (1997). "Optimal testing of VLSI analog circuits", *IEEE Trans. On Computer-aided Design of Integrated Circuits and Systems*, 16. No. 1, 58-77.

7. Efron, B. (1979). “Bootstrap methods: another look at the jackknife”, *The Annals of Statistics*, 7, 1-26.
8. Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the bootstrap*, London: Chapman and Hall.
9. Zoubir, A. M. and Boashash, B. (1998). “The bootstrap and its application in signal processing”, *IEEE Signal Processing Magazine*, January, 56-76.
10. Shannon, C. E., (1948). “The mathematical theory of communication”, *Bell Systs. Tech. J.*, 27, 379-423.
11. Shannon, C. E., (1951). “Prediction and entropy of printed English”, *Bell. Syst. Tech. J.*, 50-64.
12. Li, W. (1990). “Mutual information functions vs correlation functions”, *Journal of Statistical Physics*, 60, 823-836.
13. Z. R Yang and M. Zwolinski, “Bootstrap, an alternative to Monte Carlo simulations”, *Electronics Letters*, vol. 34, no. 12, pp1174-1175, 1998.
14. R. Battiti, “Using mutual information for selecting features in supervised neural net learning”, *IEEE Trans. on Neural Networks*, vol. 5, no. 4, pp 537-550, 1994.
15. Z. R. Yang, M. Zwolinski & C. Chalk, “Fault detection and classification in analogue integrated circuits using robust heteroscedastic probabilistic neural networks”, *4th IEEE International Mixed Signal Testing Workshop*, The Hague, The Netherlands, June 9-11, 1998, pp. 41-47.