A COMPREHENSIVE APPROACH FOR MODELING AND TESTING ANALOG AND MIXED-SIGNAL DEVICES

T. Michael Souders and Gerard N. Stenbakken

National Institute of Standards and Technology
Gaithersburg, MD 20899
(301) 975-2406

Abstract

An approach is presented for optimizing the testing of analog and mixed-signal devices. The entire process is performed with algebraic operations on an appropriate model. The paper demonstrates how this is accomplished using simple calls with public-domain software. Examples of test results achieved using this approach are included.

Introduction

Test engineers are faced with the problem of developing test routines that will correctly sort good and bad devices at minimum cost, where cost is usually reflected as test time (or throughput) together with the cost of test equipment. There are always tradeoffs to be made in terms of the completeness of the testing and the confidence that results; more exhaustive testing usually means greater confidence at the expense of throughput.

Over the last five years, a comprehensive approach has been developed at NIST for maximizing the tradeoffs associated with production testing of analog and mixed-signal devices [1-5]. The approach is based on a simple linear coefficient matrix model that relates the device response (at all candidate test conditions), to a rather small set of underlying variables.

In this paper, we assume that an accurate model is available. (Several approaches for developing the model appear in references [1,2,5]; a follow-up paper is planned that will address the modeling issue in more detail.)

Once an accurate model has been developed, simple algebraic operations on the model can be used to perform the following tasks:

1. select an optimum set of test points that will minimize the test effort and maximize the test confidence,
2. estimate the parameters of the model from measurements made at the selected test points (useful for alignment operations such as laser trimming),
3. predict the response of the device at all candidate test points (from measurements made at the selected test points) as a basis for accepting or rejecting units,
4. calculate the accuracy of the parameter estimates and response predictions, based on the random measurement error,
5. test the validity of the model, on-line, so that changes in the manufacturing process are constantly monitored, and the model can be updated.

The purpose of this paper is to explain these procedures and show how each can be performed using simple calls to routines that are available in both public domain and commercial linear algebra software packages, e.g., LINPACK, CLAM™*, and MATLAB. The routines are computationally efficient, and the most expensive ones are run off-line (and only once) for any particular device type or production run.

The approach is quite general and has been experimentally applied to the measurement of the frequency response of an amplifier-attenuator network, to fault diagnosis of a band-pass filter using time-domain measurements, and to efficient linearity tests of A/D and D/A converters. In each case, the models are unique, but they all have the same algebraic form. Therefore, the procedures of analysis are the same, and the resulting testing strategies are similarly optimized.

A Hypothetical Example

A large number of hypothetical devices, all of the same type or model, are being tested as they come off a production line. The tests consist of measurements of their behavior under a large sampling of typical and perhaps extreme input conditions. The plots of Fig. 1 show the test results - the deviations from the ideal behavior - of the first eight devices. From the test results, the devices can be sorted into performance bins. However, even though the latest automatic test equipment is being used, the tests are very time consuming to run for so many different input conditions. Is there a simpler test plan that can still accurately predict the behavior of future devices coming off the production line? The answer is usually yes.

** Certain commercial products, both software and test systems, are identified in this paper in order to adequately specify the experimental procedure. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products identified are necessarily the best available for the purpose.*

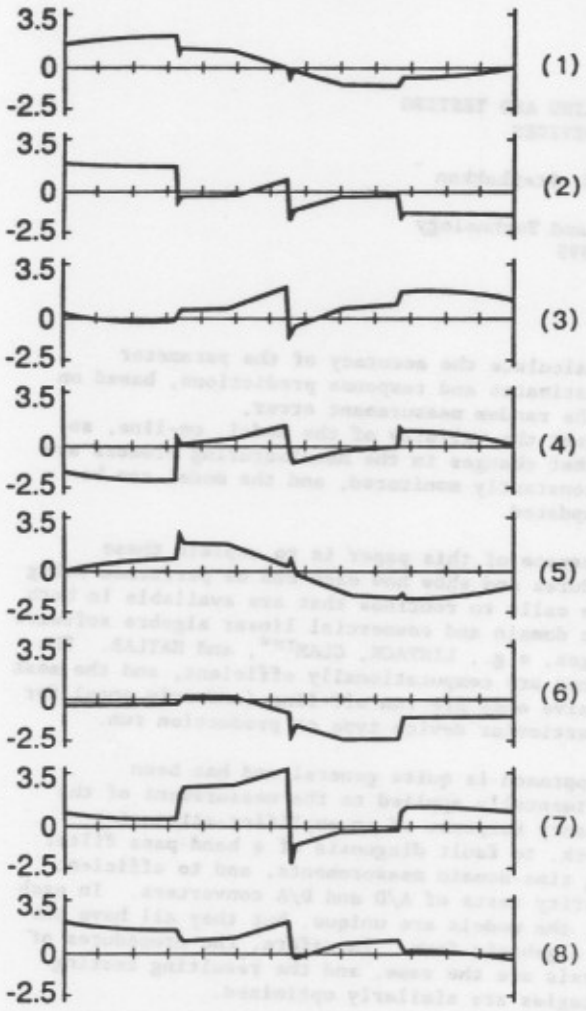

Fig. 1 Response errors vs. 128 test conditions for eight units of the hypothetical device. The scales are in arbitrary units.

In most cases, the nonideal behavior of analog or mixed signal devices is largely determined by a relatively small number of variables; in this example, the number happens to be seven. These may be thought of as the variations of a set of critical components such as resistances, capacitances, or transistor transconductances, for example, or of a set of critical process parameters such as dopant level, mask alignment, or exposure time during metalization.

Since there are only seven variables, only seven independent equations are required in order to solve the system. Therefore, the system response is actually much more constrained than might be surmised by looking at the response plots of Fig. 1. If the coefficients of the equations are known, then only seven measurements are required to completely determine the system. This is illustrated in Fig. 2. The seven curves represent the error signatures of seven variables which together can fully describe the performance of any of the devices coming off the production line. Each variable contributes more or less of its error signature, depending on its specific value. Fig. 3 demonstrates that the performance of a device from the production run can be decomposed into a weighted sum of these seven signatures; the weight of each is the value of the corresponding variable.

Each candidate test condition or test point defines a linear equation in this system; the total error at each point is the weighted sum of the seven signatures evaluated at the same test point. In this example, there are 128 separate equations, one for each candidate test point. But, as was noted above, only seven *independent* equations are needed to solve the system. This means that only seven test points actually need to be measured in order to calculate the values of the seven variables; and once the variables are known, the entire behavior of the device can be calculated at every candidate test point by appropriately weighting and summing the seven error signatures. Therefore, the test engineer really only needs to test the devices under seven conditions in order to fully characterize them - certainly a substantial time savings when compared to the full set of 128 measurements that was originally considered necessary.

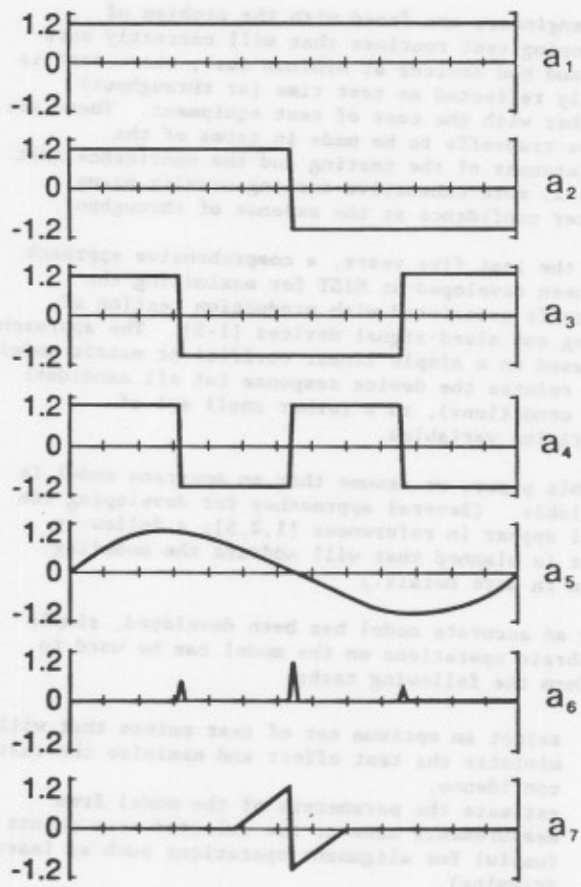

Fig. 2 Error signatures vs. 128 test conditions for the seven variables underlying the hypothetical example.

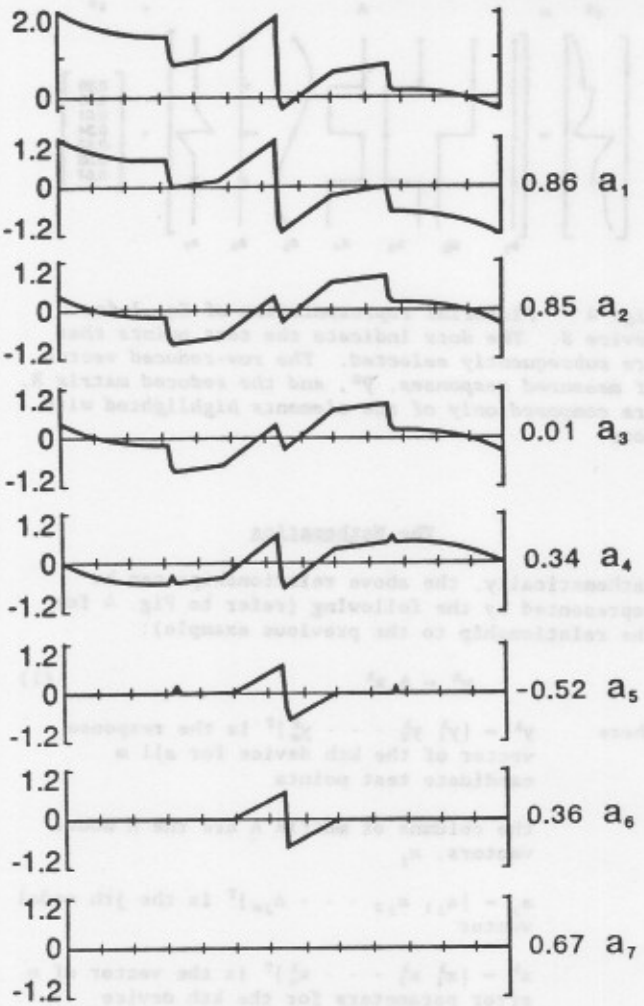

Fig. 3 Decomposition of response errors of unit 8, Fig. 1, in terms of the error signatures of Fig. 2. The top plot is the response of unit 8; the remaining plots show the errors after the designated amount of each error signature is successively subtracted.

Test Point Selection

Remember that only seven equations are needed for the example with seven variables; but also recall that the equations must be independent. Mathematically, this means that none of the seven equations can be completely expressed as a weighted sum of the remaining six. There are many different sets of seven test points that will satisfy this condition, but there are also many more that will not. In fact, there are degrees of independence as well, and what is really best is to find the set that is maximally independent; this set turns out to make the process most robust to the effects of measurement noise. For large problems, the optimal set of test points is computationally expensive to find; however, nearly optimal solutions can be found rather cheaply using an algebraic operation known as QR decomposition [3]. Computationally efficient implementations of the QRD operation are available in a number of public domain and commercial software packages [6,7,8]. These routines operate on the matrix model, i.e., the error signatures, from simple calls to the software package. The routines return a vector of the selected test points and also give information on the degree of independence represented by them. From the vector of test points, the system of corresponding simultaneous equations is known; this small system of test points and equations is valid for every device that is adequately described by the original model. The system of equations and test points is stored in memory for subsequent on-line processing as individual devices are tested.

Having made the required measurements at the selected test points, the system of equations is solved to determine the values of the seven variables, i.e., the weights to be applied to the error signatures as shown in Fig. 2. Again, standard matrix software routines are used [3]. The entire behavior of the device at all of the 128 candidate test points is now easy to predict: simply weight the seven error signatures by the corresponding values from the solution, and sum the signatures together; the result is the behavior for all test conditions, including the few that were actually measured, and the many that were not.

Barring problems with the test equipment itself, there are two sources of prediction error with this approach: measurement noise, and inadequate modeling. Measurement noise is a significant problem because it corrupts the parameter values that are estimated in the intermediate step; consequently, the predictions made from these estimated parameter values will be in error. A measure of this error is the prediction variance, and it can be computed ahead of time by knowing the original model and the selected test points [3]. (Prediction variance is the ratio of the variance of the prediction to the variance of the measurement noise. It can be evaluated at every candidate test point given a set of selected test points.) A good selection of test points is therefore one that minimizes the prediction variance. If the prediction variance is deemed too high however, even with a good selection, then it is always possible to further reduce the error by adding more test points. This will result in an overdetermined system of equations that can be solved using least squares techniques. If the additional test points again constitute a good selection, the prediction variance will be reduced by approximately the ratio of the number of test points to the number of variables.

Reducing the noise is not the only advantage of selecting more than the minimum number of test points; the redundancy permits model errors to be detected as well. By selecting additional test points, a least squares solution is found and the residuals of the solution at the test points can be generated. Examination of these can give a good indication of the accuracy of the model; a good model will produce residuals that are randomly distributed and have a standard deviation comparable to that of the measurement noise.

Modeling

There are three basic types of models that can be used in this approach: physical, *a priori*, and empirical. These will be discussed briefly to give a more complete idea of the overall approach that is being proposed; however, for the details of model development, the reader is referred to references [2-5], and to a comprehensive follow-up paper which is planned for ITC91.

Physical models are developed using simulation tools when a reasonably complete description of the device is available. These models take the form of sensitivity matrices, where the individual error signatures represent the sensitivity of the response to deviations of the device's components from their nominal values. For example, if the device's topology (i.e., connection matrix) is known, together with the nominal component values, then a simulator such as SPICE can be used to compute the sensitivity matrix. These models correspond to a first order Taylor's expansion, and are useful so long as the components of the tested devices remain reasonably close to their nominal values.

A priori models are those that use a set of vectors that are chosen to represent the device based on more general considerations. They are usually comprised of a relatively small subset (of a complete set) of basis functions that is capable of approximating the device performance to the required accuracy. For example, subsets of the Walsh functions have been used for some applications.

Empirical models are learning-based, and are obtained by numerically analyzing the data from exhaustive testing of representative units coming off the production line. They are based on the premise that a selected lot of devices will manifest all of the degrees of freedom or variability of the manufacturing process. For example, it can be shown that seven of the eight response vectors of Fig. 1 could also comprise a complete model for the device in the hypothetical example. A model can be formed from these vectors simply by substituting response vectors (1) through (7), for example, for error signatures a_1 through a_7 in the example. These models require no detailed knowledge of the internal architecture of the device or other design information; they only require a knowledge of what the expected, ideal performance should be. It will be shown in the follow-up paper that empirical models have much of the power of physical models; however, they have limited use in fault diagnosis, alignment, and trimming applications because the physical significance of the variables is often obscure.

When developing a model, it is important to ensure that the set of vectors or error signatures that comprise the model are all independent, otherwise a complete solution cannot be found. The QRD algorithm can also be used for this purpose. As in one of the examples to follow, model vectors derived using any of the above approaches can be combined to form the completed model.

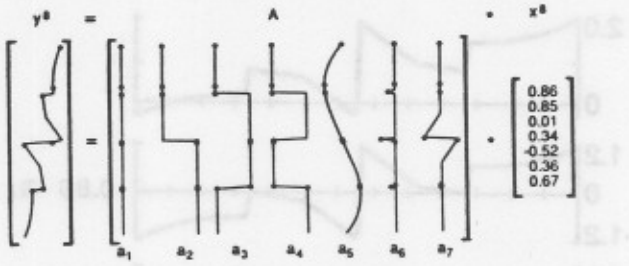

Fig. 4 Pictorial representation of Eq. 1 for device 8. The dots indicate the test points that are subsequently selected. The row-reduced vector of measured responses, $\bar{y}^s$, and the reduced matrix $\bar{A}$, are composed only of the elements highlighted with dots.

The Mathematics

Mathematically, the above relationships can be represented by the following (refer to Fig. 4 for the relationship to the previous example):

$$y^k = A\,x^k \tag{1}$$

where $y^k = [y_1^k\ y_2^k\ \cdots\ y_m^k]^T$ is the response vector of the kth device for all m candidate test points

the columns of matrix A are the n model vectors, a_j

$a_j = [a_{j1}\ a_{j2}\ \cdots\ a_{jm}]^T$ is the jth model vector

$x^k = [x_1^k\ x_2^k\ \cdots\ x_n^k]^T$ is the vector of n error parameters for the kth device (corresponding to each of the n model vectors)

and where superscript T represents the transpose operation which interchanges row and column vectors (and therefore also interchanges their subscripted indices).

Therefore, the response of the kth device at the ith test point is given by

$$y_i^k = \sum_{j=1}^{n} a_{ij}\,x_j^k = a_{i1}x_1^k + a_{i2}x_2^k + \cdots + a_{in}x_n^k \tag{2}$$

If there are n error parameters, x_j, then only n such equations are required to solve for the x_j's; i.e., a reduced system of equations is sufficient:

$$\bar{y}^k = \bar{A}\,x^k \tag{3}$$

where $\bar{}$ designates the reduced set of n test points.

As noted above, the reduced set of test points used in Eq. 3 is obtained from the full model in Eq. 1 via a call to the QRD operation. Using a standard

matrix routine, Eq. 3 is solved in terms of the reduced set of measurements:

$$\hat{x}^k = \bar{A}^{-1}\bar{y}^k \qquad (4)$$

where ^ designates variables estimated from measurement data, and $(\bullet)^{-1}$ designates the inverse (in general, Eq. 4 is solved without explicitly calculating the inverse matrix).

If more than the minimum number (n) of test points is selected for redundancy, then $\bar{A}$ is no longer square, i.e., there are more rows that columns, and Eq. 3 is solved using a least squares approach:

$$\hat{x}^k = (\bar{A}^T \bar{A})^{-1}\bar{A}^T \bar{y}^k \qquad (5)$$

In this case, the residuals of the least squares solution can be computed as

$$\epsilon^k = \bar{y}^k - \bar{A}\,\hat{x}^k \qquad (6)$$

The rms value of ϵ^k is a good measure of the accuracy of the model; it should not be appreciably larger than the measurement noise.

A good strategy for adding additional test points is to calculate the prediction variance at all candidate test points based on the test points already selected, and then select as the next test point the candidate point with the highest prediction variance. The process is repeated until the maximum prediction variance is reduced to the desired level. The normalized prediction variance can be computed as

$$\sigma_p^2/\sigma^2 = \text{diag}\,[A(\bar{A}^T\bar{A})^{-1}A^T] \qquad (7)$$

where diag [•] denotes the vector formed from the diagonal elements of [•], and σ_p^2 is the m-dimensional vector of prediction variances, and σ^2 is the variance of each measurement, assumed to be a constant.

Having estimated the variables using either Eq. 4 or 5, the entire response is predicted at all candidate test points, y^k, by performing the matrix multiplication in Eq. 1.

Mathematical Software

All of the mathematical operations described above can be performed by calls to the routines in LINPACK. LINPACK is a collection of Fortran subroutines developed to provide efficient solutions of linear systems and related problems. This is public domain software and is available free of charge or for a nominal distribution charge from a number of sources, including the original developers at the Argonne National Laboratory. A very informative users' guide [6] is also available that describes the software code and how to use the routines to efficiently solve linear system problems.

The LINPACK routines have been integrated into other mathematical software programs which add graphical routines, a convenient human interface, and automatic generation of matrix and vector variables. These integrated programs make experimentation with the modeling problems described in this paper quite easy to perform. Two such integrated programs are MATLAB and CLAM. MATLAB is available in both a public domain version and in a commercial version, and CLAM is a commercial product. Both products are available for a number of computer types. The computations done for this paper were performed on a Sun™ 3/80 workstation using CLAM software. Below is a brief description of the programs written in CLAM to solve the problems described. The MATLAB programs would be very similar.

The model matrix A is filled either with data generated by routines used to compute *a priori* vectors or the hypothetical vectors, or filled with data read from files for the empirical vectors. Similarly the measurement error vector y is filled with random numbers or data read from files of device data. The first operation in the modeling effort is to check the model vectors to make sure they are relatively independent. If they are not, the solutions can have significant errors. With the full model matrix in A, which has m rows corresponding to the candidate test points and n columns corresponding to the model parameters, use the QR function as follows:

```
QR(A,,R1,p1)
```

The matrix R1 and vector p1 are automatically generated; p1 is an n-dimensional vector giving the order in which the parameter vectors were selected and R1 is a triangular m by n matrix. (The lack of a matrix name between the commas indicates that the Q matrix will not be generated in order to minimize memory requirements.) The diagonal elements of R1 give an estimate of the condition (or degree of linear independence) of the submatrices of A, formed by adding the model vectors in the order given in p1. The diagonal of the R1 matrix is referenced as R1(0) and the condition estimate values are given by normalizing with respect to the first diagonal element R1(1,1). Thus,

```
cond1 = R1(0)/R1(1,1)
```

puts the condition estimates into the n-dimensional vector, cond1. The importance and interpretation of these numbers will be described in a subsequent paper on modeling. Suffice it to say now that the condition estimate for the last vector included in the model should not be lower than the ratio of the average of the measured error values in y to the measurement uncertainty, σ.

The initial test points are selected using the same QR function. The QR function always operates by selecting columns from a matrix. Thus, to select test points which correspond to the rows of A, use the QR function on the transpose of A. In CLAM, the transpose is designated by a prime, " ' ".

```
QR(A',,R2,p2)
```

selects the test points as the first n values of the m dimension vector p2, chosen by

```
p2r = sort(p2(1:n))
```

The sort function puts the test points into ascending order. As in the model vector selection described above, the diagonal of the R2 triangular matrix gives an estimate of the condition of the submatrices formed by adding the rows of A in the order given in p2. In general, if a model with a good condition estimate was selected in the first operation, the condition estimates for the test point selection will be good for all the n points selected. The reduced model matrix, Ar, (called A in the previous section) is chosen by

```
Ar = A(p2r,:)
```

This selects all the columns, designated by the colon, and the selected test points or rows designated by p2r.

The prediction variance for the reduced model gives the expected variance for all candidate test points relative to the measurement variance, σ^2, at the selected test points. The prediction variance as described in the previous section is the diagonal of a large matrix. The diagonal elements alone can be calculated with the following code:

```
Temp = A/(Ar' * Ar)
Temp = Temp .* A
pv = sum(Temp, #column)
```

where the matrix, Temp, is used for temporary storage, the " * " designates matrix multiplication, the " / " represents right matrix division, the " .* " represents element-by-element matrix multiplication, and the sum(·,#column) designates the sum over the columns of a vector or matrix. pv is an m-dimensional vector of the prediction variance for the reduced model Ar, i.e., for the selected test points corresponding to the rows of Ar, and for the model, A.

Additional test points can be selected that will minimize the prediction variance by first selecting the test point corresponding to the largest prediction variance in pv. The row corresponding to this test point is added to Ar to get a new model Ar. As above, the prediction variance for the new reduced model is calculated and the new largest prediction variance used to designate the next additional test point. This process is repeated to add as many test points as desired to the n original test points selected by the QR function. Note that the same test point may be selected more than once with this procedure. This means that the test point is to be measured independently more than once, so that the measurement uncertainty at that point is reduced. If this is not desired, the routine can be modified so that test points are only selected once.

With the complete number of selected test points, t, and the corresponding reduced model matrix determined (still referred to here as pr and Ar respectively), this concludes the preproduction operations. These operations are performed only once for each production device or type. For each device k measurements are made at the selected test points and subtracted from the nominal or ideal performance to give the reduced measurement error vector, yrk, for device k. The model parameters for that device, xk, are estimated with the following code

```
xk = Ar\yrk
```

where " \ " represents left matrix division. The estimated model parameters are used to generate the predicted errors, ypk, for all m candidate test points with

```
ypk = A * xk
```

If the number of test points, t, is greater than the minimum number, n, then the predicted errors at the measured test points are not the same as the measured values. The difference or residue is useful for estimating the capability of the model to predict the performance of device k. The residue for the k'th device, res_k, is calculated with

```
res_k = ypk(pr) - yrk
```

The root-mean-square of this vector, rms_k, is given by

```
rms_k = sqrt(sum(res_k .* res_k, #column)/t)
```

where sqrt(·) gives the square root of a variable, sum(·, #column) gives the sum of a column vector, and t is the total number of test measurements. To indicate a satisfactory fit of device k to the model, the value of rms_k should be of the order of the measurement uncertainty, σ.

Test Examples

These methods have been applied experimentally to several different analog and mixed signal testing problems. Two examples will be presented here: integral nonlinearity (INL) testing of an A/D converter, and time domain testing and fault diagnosis of a bandpass filter. The first example illustrates the savings in test time that can be achieved in a practical mixed-signal application, and the second example shows how the approach can be used to optimize circuit testability.

A/D Converter

As part of a cooperative effort with Teradyne, Inc., the approach was applied to 128 units from two production runs of a 13-bit, self-calibrating A/D converter. The INL tests of the devices were performed by Teradyne staff using a Teradyne model A520 test system. An 18-parameter model of the device-type was developed using a combination of *a priori* and empirical modeling techniques. The empirical modeling was performed using exhaustive test data (all 8192 codes) from the first 50 units. From the model, 18 test points (codewords) were selected using QRD, supplemented with an additional 46 selected to minimize the maximum prediction variance, for a total of 64 test codewords. The 78 devices that were not used in the development of the model were then also tested at all codes. From the all-codes data for each device, the data corresponding to the selected codes was used to

estimate the parameters of the model (Eq. 5) and the parameter estimates were used in turn to predict the response at all codes (Eq. 1). The predicted errors were then compared to the measured errors at all codes. Typical test results are plotted in Fig. 5. The top plot shows the measurement data at the 64 selected codes, the middle plot shows the predicted results for all codes, and the bottom plot shows the errors in the prediction, i.e., the difference between the measured and predicted results at all codes. The rms and peak prediction errors are 0.024 and 0.083 least significant bit (lsb), respectively.

Converters such as these are typically sorted according to their maximum INL. In Fig. 6, the error in predicting the maximum INL is plotted for 77 of the 78 devices; for comparison, the dashed lines represents the effective noise level in the measurement process, obtained by taking the standard deviation of repeated measurements of the same device. A positive error indicates the predicted maximum is smaller than the measured maximum. It can be seen that the sorting error rate based on the limited, 64-point test would be of the same order as that achieved using conventional all-codes testing involving 8192 codes. (The 0.016 lsb positive bias in this plot is likely the result of measurement noise in the all-codes data which would tend to increase the measured peaks.)

The device not included in Fig. 6 was defective. This condition was flagged during the test from the computation of model error using Eq. 6; the rms of the residuals of the least squares fit was 0.522 vs. a preset bound of 0.04. Even so, the predicted maximum for this device of 6.04 lsb's was reasonably close to the measured maximum of 5.36 lsb's.

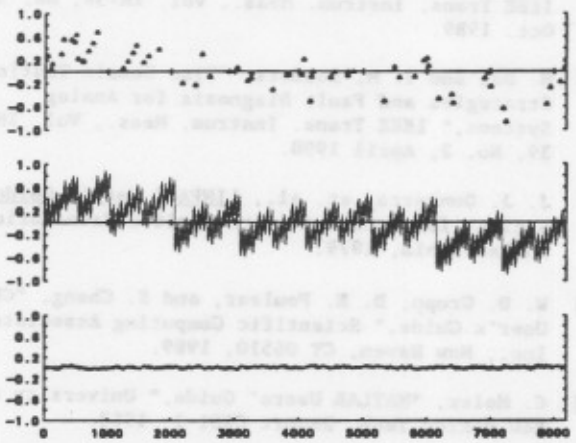

Fig. 5 Test results on one unit of the 13-bit A/D Converter. The top plot shows the INL errors measured at the 64 selected codes. The middle plot gives the predicted errors at all 8192 codes based on the 64 measurements, and the bottom plot gives the error in the prediction, i.e., the difference between the measured errors and the predicted errors at all codes. Vertical scales are in lsb's.

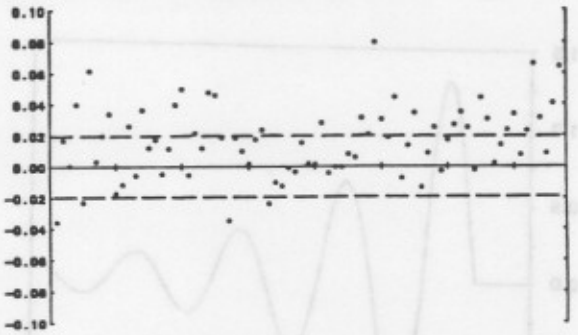

Fig. 6 Differences between the predicted and measured maximum INL for the 77 devices tested. For comparison, the dashed lines give the standard deviation of the measurement process, i.e., the repeatability of a measurement. Vertical scale is in lsb's.

Bandpass Filter

In this example, the bandpass filter shown in Fig. 7 was tested in the time domain using a step waveform as the input signal, and a waveform recorder to measure the response at selected test nodes. The objective was to accurately estimate the values of the seven components from the measured response at the test nodes; the estimates could be used to identify the components that were out of tolerance so that the circuit could be trimmed to bring it within specifications. The model consisted of a time domain sensitivity matrix computed from the system equations and connection matrix [5].

With measurements made only at the output (node 5), only three of the model vectors were found to be linearly independent, indicating that all of the components' error signatures (or sensitivity

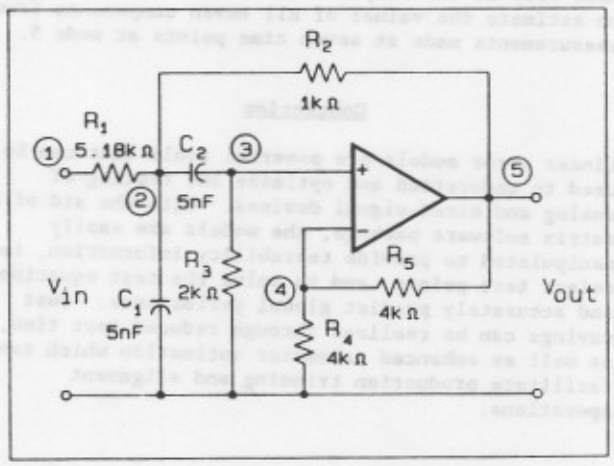

Fig. 7 Model for bandpass filter with center frequency of 24.5 kHz.

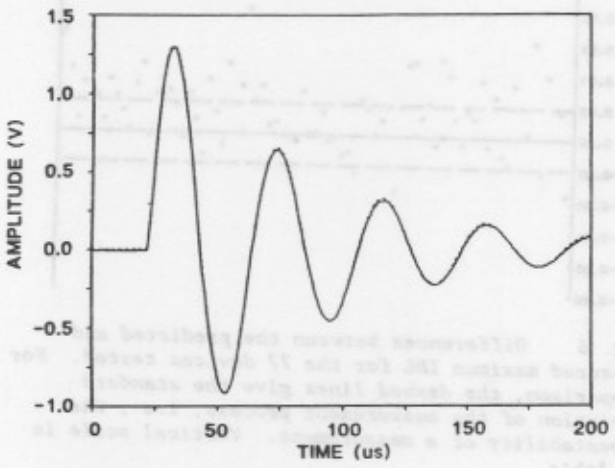

Fig. 8 Measured (solid) and predicted (dashed) step response of filter. The predicted response was based on measurements at three time points.

vectors) are not unique at this node. This however, also means that only three parameters and measurements at three time points are needed in order to predict the entire time domain output response of the circuit to an input step function. This is illustrated in Fig. 8 where the measured response is compared to the response predicted from measurements at only three time points.

Further analysis of the model showed that only five of the seven components have unique error signatures when all five nodes are measured; components R_4 and R_5 have the same signatures at all of the nodes, as do components C_1 and C_2. However, it was found that by inserting two additional resistors of known value between nodes 4 and 5, and nodes 2 and common, and remeasuring the response, the signatures of all seven components become unique with measurements made only at the output. Therefore, it is possible to estimate the values of all seven components from measurements made at seven time points at node 5.

Conclusion

Linear error models are powerful tools that can be used to understand and optimize the testing of analog and mixed-signal devices. With the aid of a matrix software package, the models are easily manipulated to provide testability information, to select test points, and to solve the test equations and accurately predict global performance. Cost savings can be realized through reduced test time, as well as enhanced parameter estimation which can facilitate production trimming and alignment operations.

Of course, not all testing problems warrant the use of these methods. Additional overhead costs are incurred in the pretest stage (when the model is developed and test points are selected), and in some cases, the computing capability must be upgraded to efficiently handle the additional computing load. These costs must be recouped in saved test time. Finally, linear error models may not be particularly well suited for some types of highly nonlinear devices.

Acknowledgement

The authors would like to thank the staff of Teradyne, Inc., and particularly Michael Amirault, for their help in obtaining and testing the devices. used in our first example, and for many valuable discussions concerning this work.

References

[1] G. N. Stenbakken, et. al., "Efficient Calibration Strategies for Linear, Time-Invariant Systems," in Proc. 1985 AUTOTESTCON Conf. (Long Island, NY), Oct. 1985, pp. 361-366.

[2] T. M. Souders and G. N. Stenbakken, "Modeling and Test Point Selection for Data Converter Testing," in Proc. 1985 Int. Test Conf. (Philadelphia, PA), 1985, pp. 813-817.

[3] G. N. Stenbakken and T. M. Souders, "Test Point Selection and Testability Measures Via QR Factorization of Linear Models," IEEE Trans. Instrum. Meas., Vol. IM-36, No. 2, June 1987.

[4] G. N. Stenbakken, T. M. Souders and G. W. Stewart, "Ambiguity Groups and Testability, IEEE Trans. Instrum. Meas., Vol. IM-38, No. 5, Oct. 1989.

[5] H. Dai and T. M. Souders, "Time Domain Testing Strategies and Fault Diagnosis for Analog Systems," IEEE Trans. Instrum. Meas., Vol. IM-39, No. 2, April 1990.

[6] J. J. Dongarra, et. al., LINPACK User's Guide, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[7] W. D. Gropp, D. E. Foulser, and S. Chang, "CLAM User's Guide," Scientific Computing Associates, Inc., New Haven, CT 06510, 1989.

[8] C. Moler, "MATLAB Users' Guide," University of New Mexico Tech. Report CS81-1, 1982.