Applying A Robust Heteroscedastic Probabilistic Neural Network to Analog Fault Detection and Classification

Zheng Rong Yang, Mark Zwolinski, *Member, IEEE*, Chris D. Chalk, and Alan Christopher Williams, *Member, IEEE*

***Abstract*—The problem of distinguishing and classifying the responses of analog integrated circuits containing catastrophic faults has aroused recent interest. The problem is made more difficult when parametric variations are taken into account. Hence, statistical methods and techniques such as neural networks have been employed to automate classification. The major drawback to such techniques has been the implicit assumption that the variances of the responses of faulty circuits have been the same as each other and the same as that of the fault-free circuit. This assumption can be shown to be false. Neural networks, moreover, have proved to be slow. This paper describes a new neural network structure that clusters responses assuming different means and variances. Sophisticated statistical techniques are employed to handle situations where the variance tends to zero, such as happens with a fault that causes a response to be stuck at a supply rail. Two example circuits are used to show that this technique is significantly more accurate than other classification methods. A set of responses can be classified in the order of 1 s.**

***Index Terms*—Automatic test pattern generation (ATPG) testing, fault-diagnosis, quiescent supply current (IDDQ), mixed-signal_test.**

I. Introduction

THERE has been much interest in recent years in the testing of analog integrated circuits and in the development of appropriate testing techniques. Such work has concentrated on the detection of catastrophic faults, such as opens and shorts. Detection of faults has depended on the responses of faulty circuits being *sufficiently* different from the fault-free response. The definition of sufficiency has varied from arbitrary thresholds of, say, 10% through to differences of the order of 100% or more.

All integrated circuits, fault-free or otherwise, are subject to parametric changes due to process variations. It is reasonable to assume that process parameter variations of up to 3σ from their nominal values will not cause an otherwise fault-free circuit to fail. Indeed, circuits are often designed to be tolerant of such variations. Equally, it is reasonable to assume that the process parameters of a circuit containing a catastrophic fault can also vary by up to 3σ. Therefore, a further definition of detectability is that the faulty and fault-free responses differ by at least 6σ [1]. It has, however, also been noted that the distribution of the responses of faulty circuits about the nominal value is not necessarily the same as that of the fault-free circuit [1]. An extreme example of this is the effect of a fault that causes an output to be stuck at a supply rail, regardless of parametric variations; in such a case, the variance of the faulty response would be zero. Even if the distributions of both the fault-free and faulty responses are Gaussian, the variances of the faulty responses are almost certainly not the same as that of the fault-free circuit. Moreover, the assumption of a Gaussian distribution is questionable [2].

Determination of a suitable detection threshold for a given test, therefore, relies upon Monte Carlo simulations of the fault-free circuit, with respect to parametric variations, and of all the possible faulty circuits. This is extremely laborious.

It has been suggested, therefore, that simple thresholds are insufficient for fault detection and statistical methods should be employed [3]. Linear discriminant analysis (LDA), which is a classical statistical method [4], was, therefore, applied in 1993 [3]. Because of the limitations of LDA, Epstein suggested that neural networks should be applied to the area of fault detection and classification [3]. Since then, neural networks have been applied in [5]–[8]. Generally speaking, the use of neural networks as discriminators is at a very early stage with limitations such as small circuit size [5], [6] nonrobustness [7] and nonquantitative prediction [8]. In this paper, we employ a novel type of neural network, which can give very accurate fault detection and classification for larger circuits. Furthermore, the method developed in this paper automatically clusters faults. When a novel pattern is fed into the model, the model can tell us two important pieces of information. One is the probability of the circuit being faulty and the other indicates the type of fault.

In Section II, we discuss analogue fault detection, diagnosis, and statistical classification. We then simply describe the structure of the novel neural network employed in this paper. We follow this with two analog circuit examples that compare the use of this neural network technique to probabilistic thresholds and with simpler neural network methods.

Manuscript received July 21, 1998; revised July 15, 1999. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant GR/L35829.

Z. R. Yang is with the Department of Physics, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, Scotland, U.K.

M. Zwolinski is with Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.

C. D. Chalk is with Philips Semiconductors Ltd, Millbrook Ind. Est, Southampton, SO15 0DJ, U.K.

A. C. Williams is with Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K.

Publisher Item Identifier S 0278-0070(00)01366-X.

II. Analogue Fault Detection, Diagnosis vs Statistical Classification

A. 3σ Detection Threshold

The objective of analog fault detection or diagnosis is to identify whether an analog circuit is faulty or not by comparing n circuit measurements, $\mathbf{x}_m \in \Re^n$, with n circuit specifications, $\mathbf{x}_s \in \Re^n$. Fault detection or diagnosis needs to map the comparison to a binary or discrete numerical value

$$\mathbf{x}_m \sim \mathbf{x}_s \rightarrow o \in \Re^1. \tag{1}$$

Analog fault detection and diagnosis, therefore, involves numerical calculation. Fault detection and diagnosis have different objectives: fault detection is commonly used in the production test of IC's, while fault diagnosis can be used in circuit design [9]. $\mathbf{x}_s$ can be regarded as a benchmark—the response of fault-free circuit. The 3σ method [1] is one example

$$\mathbf{x}_m \sim \mathbf{x}_s = \begin{cases} \text{faulty}, & \text{if } \|\mathbf{x}_m \sim \mathbf{x}_s\| > 3\sigma_s \\ \text{fault-free}, & \text{otherwise.} \end{cases} \tag{2}$$

For the problem of fault diagnosis, the benchmarks are usually obtained by fault simulation. After fault simulation, faults are clustered into several groups, which constitute the benchmarks for fault diagnosis.

B. Sensitivity-Based Methods

The 3σ method and the statistical and neural network methods, described below, require a large amount of simulation or experimental data. Sensitivity-based methods can be used for fault analysis and diagnosis [10]. The sensitivity of an output response to any parametric change can be calculated using just two circuit analyzes, one of the original circuit and one of the adjoint network [11]. All sensitivity methods assume linearity in the circuit, however. In particular, it is assumed that the insertion of a fault will not alter the operating point of the circuit. In general, this assumption cannot be guaranteed, although a large number of special cases exist where that assumption holds. Because sensitivity-based methods are not generally applicable, we do not consider them further in this paper.

C. LDA

Fault detection or fault diagnosis is an application of multivariate statistical classification, which aims to quantify a multidimensional data space [12]. Classical multivariate statistical classification technologies include LDA, quadratic discriminant analysis and logit analysis [12]. Of these, LDA is the most rigorous and general technology [4]. LDA was applied to the fault detection and diagnosis problem in 1993 [3]. There are no reports of applying quadratic discriminant analysis or logit analysis to fault detection and diagnosis. LDA requires multivariate normally distributed variables and the same dispersion (variance-covariance matrices). In contrast to regression analysis, the coefficients of the selected variables are not uniquely defined. The technique decides which variables to include, by maximizing the ratio of the "between groups" to "within groups" variance. The output of the function or model is a binary variable, which indicates whether an observation belongs to one class or another. LDA is formalized as

$$\mathbf{w} \cdot \mathbf{x}_m \sim \mathbf{w} \cdot \mathbf{x}_s \rightarrow o \in \Re^1 \tag{3}$$

where $\mathbf{w}$ is the linear mapping vector [12]. As mentioned above, the distribution of a faulty population rarely meets this assumption.

D. Neural Networks

Neural networks mimic intelligence [13]. The learning or training mode of neural networks is different from that of traditional statistical methods. The latter attempt to give a rigid description of the underlying functions, while neural networks do not force each element in a network to be rigid and the whole system can have some degree of tolerance, so that the ability to generalize (in terms of testing or prediction) can be increased. Generally speaking, there are four types of neural network:

- back-propagation neural network (BPNN) [26];
- probabilistic neural network (PNN) [19], [20];
- self-organizing mapping (SOM) [13]; and
- radial basis function neural network (RBF) [28], [29].

Yang and Musgrave first applied neural networks to analog circuit fault detection and diagnosis in 1993 [5], where BPNN and SOM were both utilized for the purpose of fault detection and diagnosis. Yu *et al.* applied the back propagation neural network to the fault diagnosis of a CMOS opamp circuit with 11 MOSFET's [6]. Nine gate oxide short faults for each MOSFET were inserted. Therefore, there were 100 fault patterns (nine faults $*$ 11 MOSFET's + one fault-free). Each pattern contained 20 dimensions (sample points). The diagnostic accuracy of the training patterns reported by Yu *et al.* were 67% and 83.3% for ramped and sinusoidal test stimuli, respectively. Yu's work was further extended for multiple fault diagnosis in [7].

There are four drawbacks to these approaches. The first problem is that the BPNN generally requires a large number of training patterns to let the network learn the underlying mapping function (mapping a data space which contains faulty and fault-free responses to a desired diagnosis space). Only a network trained in this way can have enough tolerance for real diagnosis. The work described in [6] employed only one pattern for each class, thus a network trained by the data set employed in [6] will inevitably be impractical. The second problem is that the accuracy of the training patterns should not be a measure of whether a model is good or not. However, the model developed in [6] was measured by its accuracy with respect to the training patterns alone. The third problem is that the population of faulty circuits is much larger than the population of fault-free circuits. Using such an unmatched data set, the BPNN is usually unable to constitute a correct classifier, because a classifier generated by a BPNN based on this type of data set will have a large difference between the rate that misclassifies faulty circuits and the rate that misclassifies fault-free circuits. This problem has been intensively studied in [14] and [15]. Yang and Musgrave [5] attempted to overcome this problem by over-sampling the fault-free circuit, which makes the data set heavier and can cause an unnecessarily large computational cost. The fourth problem is that BPNN is prone to over-fit if no validation data set is employed. However, the work conducted in [6] and [7] did not employ validation data at all. Over-fitting (also called over-learning) is a phenomenon, whereby a model has good performance for the data used for the model constitution, but has low reliability with novel data [16].

SOM was used to cluster circuit faults in 1996 [8]. SOM is known as a topological mapping algorithm, in which patterns with similar characteristics cluster together automatically. Output nodes will thus be ordered by competitive learning. SOM shows good performance in visualization, but the learning rate and the neighbor size of SOM have to be optimally selected by experience and a SOM net needs a long time to converge [17]. Furthermore, it is difficult to determine the boundary on SOM mapping space for diagnosis in practice and SOM is unable to give a quantitative analysis [15].

There are no reports of applying RBF and PNN to fault detection and diagnosis to date. RBF is a feed-forward neural network. Although there have been some studies on how to select the basis function for RBF, such as the OLS algorithm [18], optimally selecting the basis function for the classification problem is not easy and most algorithms involving RBF tend to employ the same variance for all the basis functions. This is questionable in real applications.

III. The Robust Heteroscedastic Probabilistic Neural Network

A PNN classifies data by estimating the class conditional probability density functions [19], [20] because the parameter (the common variance) of a PNN cannot be determined analytically. To do this it requires a *training* phase, followed by a *validation* phase, before it can be used in a testing *phase*. A PNN consists of a set of Gaussian kernel functions. The original PNN uses all the training patterns as the centers of the Gaussian kernel functions and assumes a common variance or covariance (*homoscedastic* PNN). To avoid using a validation data set and to determine analytically the optimal common variance, a maximum likelihood (ML) procedure was applied to PNN training [23]. However, Streit's PNN [23] is still homoscedastic. On the other hand, the Gaussian kernel functions of a *heteroscedastic* PNN are uncorrelated and separate variance parameters are assumed. This type of PNN is more difficult to train using the ML training algorithm because of numerical difficulties [15]. A robust method has been proposed to solve this numerical problem by using the *jackknife*, a robust statistical method [25], hence the term "robust heteroscedastic probabilistic neural networks" (RHPNN).

The RHPNN is a four layer feedforward neural network based on the Parzen window estimator [21] that realizes the Bayes classifier given by (4)

$$g_{Bayes}(\mathbf{x}) = \arg(\max_{1\le j\le K}\{\alpha_j f_j(\mathbf{x})\}) \tag{4}$$

where $\mathbf{x}\in\Re^d$ is a d-dimensional pattern, $g(\mathbf{x})$ is the class index of $\mathbf{x}$, the *a priori* probability of class j $(1\le j\le K)$ is α_j and the conditional probability density function of class j is f_j. The object of the RHPNN is to estimate the values of f_j. This is done using a mixture of Gaussian kernel functions.

The first layer of the PNN is the input layer. The second layer is divided into K groups of nodes, one group for each class. The ith kernel node in the jth group is described by a Gaussian function (5)

$$p_{i,j}(\mathbf{x}) = \frac{1}{(2\pi\sigma_{i,j}^2)^{d/2}}\exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_{i,j}\|^2}{2\sigma_{i,j}^2}\right) \tag{5}$$

where $\mathbf{c}_{i,j}\in\Re^d$ is the mean vector and $\sigma_{i,j}^2$ is the variance. The third layer has K nodes; each node estimates f_j, using a mixture of Gaussian kernels, from (6)

$$f_j(\mathbf{x}) = \sum_{i=1}^{M_j}\beta_{i,j}p_{i,j}(\mathbf{x}) \qquad 1\le j\le K \tag{6}$$

where M_j is the number of nodes in the jth group in the second layer; and $\beta_{i,j}$ satisfies (7)

$$\sum_{i=1}^{M_j}\beta_{i,j} = 1, \qquad 1\le j\le K. \tag{7}$$

The fourth layer of the PNN makes the decision from (4). The PNN is heteroscedastic when each Gaussian kernel has its own variance. The centres, $\mathbf{c}_{i,j}$, the variances, $\sigma_{i,j}^2$ and the mixing coefficients, $\beta_{i,j}$ have to be estimated from the training data. One assumption is

$$\alpha_j = \frac{1}{K} \qquad 1\le j\le K. \tag{8}$$

The EM algorithm [22] has been used to train homoscedastic PNN's [23]. Each iteration of the algorithm consists of an expectation process (E) followed by a maximization process (M). This algorithm converges to the ML estimate. For the heteroscedastic PNN, the EM algorithm frequently fails because of numerical difficulties. These problems have been overcome by using a *jackknife*, which is a robust statistical method [25].

Suppose the training data is partitioned into K subsets, $\{\mathbf{x}_n\}_{n=1}^N = \{\{\mathbf{x}_{n,j}\}_{n=1}^{N_j}\}_{j=1}^K$, where $\Sigma_{j=1}^K N_j$ is the total number of samples and N_j is the number of training samples for class j. The training algorithm is now expressed as follows, where $\tilde{\sigma}_{m,i}^2|^{(k)}$ and $\tilde{\mathbf{c}}_{m,i}|^{(k)}$ are the jackknife estimates of the previous values of $\sigma_{m,i}^2$ and $\mathbf{c}_{m,i}$, respectively.

Step 1: Compute weights for $1\le m\le M_i$, $1\le n\le N_i$ and $1\le i\le K$.

$$w_{m,i}^{(k)}(\mathbf{x}_{n,i}) = \frac{\beta_{m,i}p_{m,i}^{(k)}(\mathbf{x}_{n,i})}{\sum_{l=1}^{M_i}\beta_{l,i}p_{l,i}^{(k)}(\mathbf{x}_{n,i})} \tag{9}$$

where

$$p_{l,i}^{(k)}(\mathbf{x}_{n,i}) = \frac{1}{(2\pi\tilde{\sigma}_{l,i}^2|^{(k)})^{d/2}}\exp\left(-\frac{\|\mathbf{x}_{n,i}-\tilde{\mathbf{c}}_{l,i}|^{(k)}\|^2}{2\tilde{\sigma}_{l,i}^2|^{(k)}}\right). \tag{10}$$

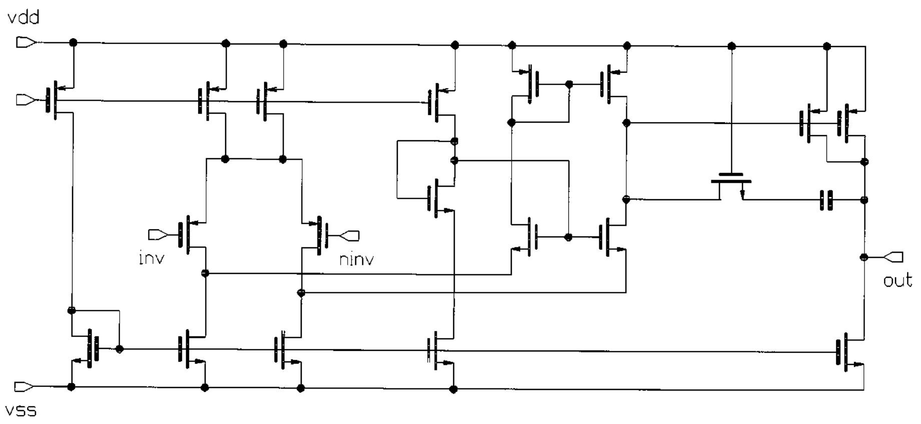

Fig. 1. Folded Cascode Opamp, used for Case 1.

Step 2: Update the parameters for $1 \leq m \leq M_i$ and $1 \leq i \leq K$

$$\tilde{\mathbf{c}}_{m,i}|^{(k+1)} = N_i \mathbf{c}_{m,i}|^{(k+1)} - \frac{N_i - 1}{N_i} \sum_{j=1}^{N_i} \mathbf{c}_{m,i}|_{-j}^{(k+1} \quad (11)$$

where

$$\mathbf{c}_{m,i}|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})} \quad (12)$$

$$\mathbf{c}_{m,i}|_{-j}^{(k+1)} = \frac{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\mathbf{x}_{n,i}}{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})} \qquad 1 \leq j \leq N_i \quad (13)$$

Similarly,

$$\tilde{\sigma}_{m,i}^2|^{(k+1)} = N_i \sigma_{m,i}^2|^{(k+1)} - \frac{N_i - 1}{N_i} \sum_{j=1}^{N_i} \sigma_{m,i}^2|_{-j}^{(k+1)} \quad (14)$$

where

$$\sigma_{m,i}^2|^{(k+1)} = \frac{\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\|\mathbf{x}_{n,i} - \tilde{\mathbf{c}}_{m,i}|^{(k)}\|^2}{d\sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})} \quad (15)$$

$$\sigma_{m,i}^2|_{-j}^{(k+1)} = \frac{\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})\|\mathbf{x}_{n,i} - \tilde{\mathbf{c}}_{m,i}|^{(k)}\|^2}{d\sum_{n=1,n\neq j}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i})}, \quad 1 \leq j \leq N_i \quad (16)$$

and

$$\beta_{m,i}|^{(k+1)} = \frac{1}{N} \sum_{n=1}^{N_i} w_{m,i}^{(k)}(\mathbf{x}_{n,i}). \quad (17)$$

Further details about the RHPNN are published elsewhere [24].

IV. Experimental Results

A. Case 1, Operational Amplifier Example

The operational amplifier circuit of Fig. 1 was used to evaluate the RHPNN method of fault classification. Short and open faults were modeled using 1-Ω short circuit (gate-source, gate-drain) and 1 TΩ open circuit (drain open, source open) transistor fault models. The opamp and the voltage reference (not shown) have a total of 25 transistors, giving a total of 100 possible faults. Of these, eight faults are redundant, because they duplicate other faults or because gate-source connections already exist. The stimulus was a sinusoid of 0.5 V amplitude at 300 kHz with a dc offset of –3 V. The fault-free and each faulty circuit were simulated using nominal parameter values. Thirty Monte Carlo simulations were performed on each version of the circuit to model parametric variations due to process changes. The following process parameters were varied: $Vt, Tox,$ the mobility, and the lateral diffusion. Each HSPICE simulation run took approximately 38 s on an UltraSPARC 30. The simulation times varied depending on which particular faults were modeled. The overall simulation time was 29.5 h.

From each simulation, four parameters were derived: the dc voltage at the output; the dc supply current; the rms value of the ac component of the output voltage and the rms value of the ac component of the supply current. to give an indication of the effectiveness of the RHPNN method, a simple threshold of detection was used for comparison, (18), such that a fault was deemed detectable if this condition were met [1]

$$(|\mu_{\text{faulty}} - \mu_{\text{fault-free}}| - 3.|\sigma_{\text{faulty}} - \sigma_{\text{fault-free}}|) > 0 \quad (18)$$

where σ is the standard deviation and μ is the mean.

B. Training the RHPNN

The training procedure was as follows. The results of 15 of the Monte Carlo simulations of the fault-free circuit and two of

TABLE I
FAULT GROUPS FOR TRAINED RHPNN

Fault Name	1st training result	2nd training result	Fault Name	1st training result	2nd training result
Fault-Free	0	0	xiref1_m12_mdop1	3	2
xiref1_m11_mgss1	0	1	xiref1_m12_msop1	3	2
xiref1_m5_mdop1	0	2	xiref1_m13_mdop1	4	1
xopa6_xm7_m4_mdop1	1	1	xiref1_m13_msop1	4	1
xopa6_xm7_m4_msop1	1	1	xopa6_xm3_m8_mgds1	4	1
xopa6_xm8_m1_mdop1	1	1	xopa6_xm7_m4_mgss1	4	1
xopa6_xm8_m1_msop1	1	1	xiref1_m18_msop1	4	2
xopa6_xm11_m6_mgds1	1	2	xiref1_m12_mgss1	4	3
xiref1_m17_mgss1	1	3	xopa6_xm3_m4_mgds1	4	4
xopa6_xm3_m7_mgds1	1	4	xopa6_xm6_m4_mgds1	5	1
xopa6_xm9_m1_mgds1	1	5	xopa6_xm13_m5_mdop1	5	2
xopa6_xm12_m5_mgds1	1	6	xopa6_xm3_m5_mdop1	5	3
xopa6_xi7_m4_mgss1	1	7	xopa6_xm3_m5_msop1	5	3
xopa6_xm1_m4_msop1	1	7	xopa6_xm3_m7_mdop1	5	3
xopa6_xm10_m1_mdop1	1	7	xopa6_xm3_m7_msop1	5	3
xopa6_xm10_m1_mgss1	1	7	xopa6_xm5_m4_mgds1	5	3
xopa6_xm11_m4_mdop1	1	7	xiref1_m11_mgds1	5	4
xopa6_xm11_m4_msop1	1	7	xopa6_xm13_m5_msop1	5	5
xopa6_xm13_m5_mgds1	1	7	xopa6_xm2_m4_mdop1	6	1
xopa6_xm13_m5_mgss1	1	7	xopa6_xm2_m4_msop1	6	1
xopa6_xm5_m4_mdop1	1	7	xopa6_xm10_m1_mgds1	6	2
xopa6_xm5_m4_msop1	1	7	xiref1_m8_mdop1	7	1
xopa6_xm6_m4_mdop1	1	7	xiref1_m8_msop1	7	1
xopa6_xm6_m4_msop1	1	7	xopa6_xm12_m4_mdop1	8	1
xiref1_m14_mdop1	1	8	xopa6_xm12_m4_msop1	8	1
xiref1_m17_mdop1	1	8	xopa6_xm4_m4_mgss1	8	1
xiref1_m17_msop1	1	8	xopa6_xi7_m4_mgds1	8	2
xopa6_xm3_m4_mdop1	1	8	xiref1_m5_mgss1	9	
xopa6_xm3_m4_msop1	1	8	xopa6_xi6_m4_msop1	10	1
xopa6_xm1_m4_mgds1	1	9	xopa6_xi7_m4_msop1	10	1
xopa6_xm10_m1_msop1	1	10	xiref1_m8_mgss1	10	2
xopa6_xm1_m4_mgss1	2	1	xopa6_xi6_m4_mdop1	10	3
xopa6_xm11_m6_mgss1	2	1	xopa6_xi7_m4_mdop1	10	3
xopa6_xm11_m6_msop1	2	1	xiref1_m11_mdop1	10	4
xopa6_xm12_m4_mgds1	2	1	xiref1_m11_msop1	10	4
xopa6_xm2_m4_mgss1	2	1	xiref1_m4_mdop1	10	4
xopa6_xm3_m8_msop1	2	1	xiref1_m4_msop1	10	4
xopa6_xm8_m1_mgss1	2	1	xopa6_xm12_m5_mdop1	10	4
xopa6_xm3_m8_mdop1	2	2	xopa6_xm13_m4_mdop1	10	4
xopa6_xm9_m1_mgss1	2	2	xopa6_xm13_m4_msop1	10	4
xopa6_xm9_m1_msop1	2	2	xopa6_xm12_m5_msop1	10	5
xopa6_xm2_m4_mgds1	2	3	xiref1_m10_mdop1	10	6
xopa6_xm4_m4_mdop1	2	4	xiref1_m10_msop1	10	6
xopa6_xm4_m4_msop1	2	4	xiref1_m9_mdop1	10	6
xopa6_xm11_m6_mdop1	2	5	xiref1_m9_msop1	10	6
xiref1_m10_mgss1	3	1	xiref1_m5_msop1	10	7
xiref1_m9_mgss1	3	1			

each of the Monte Carlo simulations of each faulty circuit were pooled together at random for training. At first, all the patterns in the pool are used to build up a model, which is able to group all the fault-free and faulty circuits into n groups. The strategy for selecting the value of n is to ensure each kernel has at least one pattern (of a fault-free or faulty circuit) falling in it. The optimal n in this case is 11, as shown in Table I, in which, the ninth group only contains one pattern: the fault `xiref1_m5_mgss1`. It should be noted that the RHPNN works in a different way to other neural networks, such as BPNN [26] or SOM [13]. With the RHPNN, it is not necessary to define a class label for each faulty pattern, which is a vector containing voltages, currents or both corresponding to a faulty circuit. All the faulty patterns are labeled with the same number when training a RHPNN model. During training, the RHPNN is able to cluster the patterns automatically. This is an advantage compared with other neural networks. We know that some faulty patterns are very close to the fault-free patterns and some faulty patterns have a large deviation from other faulty patterns. This phenomenon affects the training of a neural network. After the first training is completed, it is, therefore, worthwhile considering a further training phase to refine the model. The strategy of the further training is to retrain the RHPNN for those groups containing more than one faulty pattern. The method is very simple in that the new pool for retraining is composed of 15 randomly selected fault-free patterns plus the faulty patterns appearing in the group on which we are focusing.

The third column in Table I indicates the group label for the patterns after the second training while the second column has

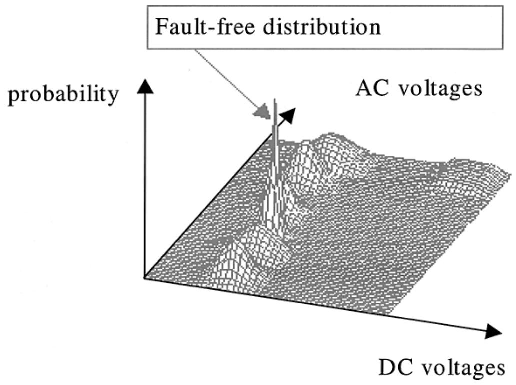

Fig. 2. Probability distribution for ac and dc voltages.

the group label for the patterns after the first training. The results in Table I were generated by a RHPNN with four dimensions, which meant using all the voltage and current measurements. The faults are described by the name of the MOSFET and its location followed by the terminals that are shorted together. Thus "`xopa6_xm10_ml_mgss1`" refers to a fault in the opamp (opa6) shorting the gate to the source of transistor m1 in subcircuit xm10.

Fig. 2 shows the probability distributions for the various groups, when the ac and dc voltages are considered. Note that the variance of the fault-free group is much smaller than that of the other groups. Similarly Fig. 3 shows the four-dimensional (4–D) probability distribution. Here, it can be seen that the probability distributions of the various groups are very distinct. It should be noted that the first and second dimensions in Fig. 3 do not signify any real measurements. These two dimensions are transferred from the Sammon mapping, a multidimensional scaling technique [27].

C. Fault Detection

Table II shows the percentage of correct classifications between the faulty and fault-free circuits using the 3-σ threshold test, the original PNN and a RHPNN for each of the parameters measured individually. Note that the Bayes method misclassifies all the fault responses in the dc voltage test.

It can be seen that the RHPNN has significantly better fault detection compared with other methods. For example, the RHPNN improves the fault detection compared with the 3-σ thresholding method to a significance level of 0.1%, except for the dc current case.

Table III shows how the accuracy of classification may be improved by pairing parameters for the threshold test, a (standard) PNN [19], [20] and a RHPNN, where "AC" is the data space formed by ac voltages and ac currents; "DC" is the data space formed by dc voltages and dc currents; "Voltage" is the data space formed by ac voltages and dc voltages; and "Current" is the data space formed by ac currents and dc currents. Similarly, Table IV shows the fault detection rate when all four parameters are used.

D. Misclassifications

Table V gives the details of misclassification when applying the RHPNN to this example. It can be seen that only 1 fault-free

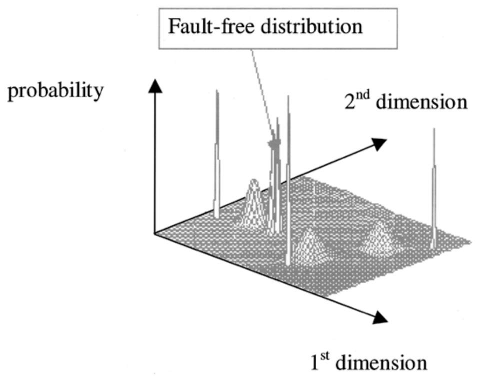

Fig. 3. Probability distribution for the 4-D case.

TABLE II
CORRECT CLASSIFICATION OF FAULTY CIRCUITS USING THRESHOLD, PNN, AND RHPNN METHODS

Method	AC(voltage)	AC(current)	DC(voltage)	DC(current)
Threshold	49%	61%	55%	58%
PNN	82%	70%	57%	2%
RHPNN	79%	77%	71%	60%

TABLE III
CORRECT CLASSIFICATION OF FAULTY CIRCUITS IN TWO DIMENSIONS USING THRESHOLD, PNN, AND RHPNN

Method	AC	DC	Voltage	Current
Threshold	63%	66%	62%	72%
PNN	71%	42%	69%	5%
RHPNN	96%	80%	96%	78%

TABLE IV
CORRECT CLASSIFICATION IN FOUR DIMENSIONS USING THRESHOLD, PNN, AND RHPNN

Method	
Threshold	73.00%
PNN	71.97%
RHPNN	99.96%

TABLE V
MISCLASSIFIED FAULT-FREE AND FAULTY CIRCUITS

	Fault-free	Faulty	Percentage
Fault-free	12	1	92.31%
Faulty	0	2584	100%
Total			99.96%

circuit is mis-detected and none of the faulty circuits is mis-detected (the fault coverage is 100%). The hypothesis that the population of misclassified fault-free circuits and the population of misclassified faulty circuits are from two different populations is rejected by the t-test [12]. This means that the two populations are from the same population or that there is no significant

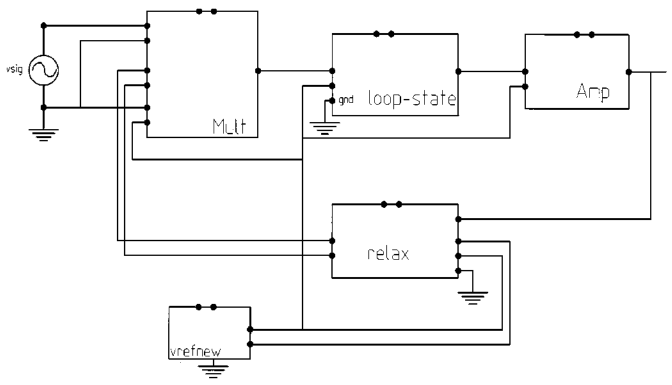

Fig. 4. PLL circuit structure.

difference between the rate that misclassifies fault-free circuits and the rate that misclassifies faulty circuits.

E. Case 2, Phase-Locked Loop Example

The second example considered here is a PLL circuit. The circuit structure is shown in Fig. 4. The circuit consists of an analog multiplier, filter (one pole and one zero), amplifier and voltage controlled oscillator (VCO). The circuit contains 122 transistors. Three-hundred-ninety-two unique open and short faults were inserted using the same transistor fault model as in Section IV-A. The circuit was stimulated using a 0.8 MHz sine wave, with an amplitude of 0.5 V. This frequency is in the center of the locking range for the PLL and is, therefore, a suitable test of the normal operation of the PLL. Table VI lists the measurements made from the faulty and fault-free simulations. The fault-free circuit was simulated with 30 Monte Carlo runs, while each faulty circuit was simulated by ten Monte Carlo runs. Each simulation took around 62 s on an Ultra SPARC 30; the total simulation time was, therefore, around 68 h. Excluding non-convergent faults, we obtained 3490 samples from these simulations.

The 3490 samples were divided into two subsets: a training data set and a testing data set. The training was done using 22 runs. Of these 22 runs, six runs were for the original tests (one dimension), shown in Table VI, 15 runs were for arbitrary combinations of any two of the original tests (two dimensions) and one run was the combination of all six original tests (six dimensions). The results in Figs. 5 and 6 show that the fault detection using the combination of dc and Idd tests has the lowest misclassification rate (0%) when using the robust heteroscedastic probabilistic neural network. Fig. 5 compares the RHPNN and 3-σ methods. It can be seen that the RHPNN gives a significant improvement over the 3-σ method for all the runs. Fig. 6 shows the comparison between the RHPNN and PNN methods. In general, the RHPNN is better than the PNN, but the advantage is not as marked as for the first example.

It is interesting to note that the dc output voltage tests combined with the dc supply current tests (dc + idd) gave the highest fault coverage, 100%. We observed that many of the faults caused the output to be stuck at the supply rail (5 V) with zero variance and so are easy to detect with these dc tests. It also appears that the dc output voltage and dc supply current tests are complimentary. Moreover, these dc structural tests

TABLE VI
MEASUREMENTS FOR STRUCTURAL TESTING

Name	Meaning
AC	RMS value of the AC component of the output voltage.
Avg	Average value of the output voltage.
DC	RMS value of the output voltage.
FC	Lock frequency.
Idd	DC power supply current (Vdd).
Vdd	RMS value of the AC component of the power supply current.

have higher fault coverages than a specification test: the lock frequency (FC).

F. Run Time Behavior of RHPNN

Table VII shows the time taken to train the RHPNN for the Opamp example and the time taken to apply a test using the RHPNN to the test data. The HSPICE simulation time is not included in these figures. These results were obtained on an UltraSPARC 30 running at 300 MHz. It can be seen that pass/fail decisions on all the circuits can be made in the order of 1 s. Note also that the training time for a new circuit is typically of the order of 1 min.

Fig. 7 gives the negative log likelihood function at each iteration step for training a data space formed by all four parameters using RHPNN. When a system is stable, the negative log likelihood function will asymptote to a constant. From Fig. 7, it can be seen that 20 steps are needed in this case.

Table VIII compares the run-time behavior of the RHPNN, PNN and BPNN. In Table VIII, the number of converging steps for the PNN is determined by the step increment, which is defined by the user. From Table VIII, it can be seen that the BPNN is much more time consuming than the PNN and RHPNN. The PNN needs less time for training because there is no need for ML training of the parameters of the neural network. However, it needs a longer test time because it uses all the training patterns as kernels. In this application, the PNN employs all the 201 training patterns as kernels, whilst RHPNN only uses 20 kernels.

Table IX shows the run time behavior of the RHPNN for the PLL example for the combination of dc and Idd tests. It can be seen that after only 30 steps, the system is stable. It should be noted that there are total of 2784 samples selected for testing. The post-test time required for one circuit is only about 0.02 s (53/2784).

The HSPICE simulation times are significantly greater than the times taken by the RHPNN to classify the simulation data. Although this may appear to be a disadvantage of the technique, it should be recalled that only 15 Monte Carlo simulations of the fault-free circuit and 2 Monte Carlo simulations of each of the faulty circuits were used to *train* the RHPNN. Therefore, the simulation times required to generate data for *training* the RHPNN are around 2.1 h and 13.8 h for the opamp and the PLL, respectively. No further simulation would be required if the RHPNN thus generated were to be used for testing real devices as they were manufactured.

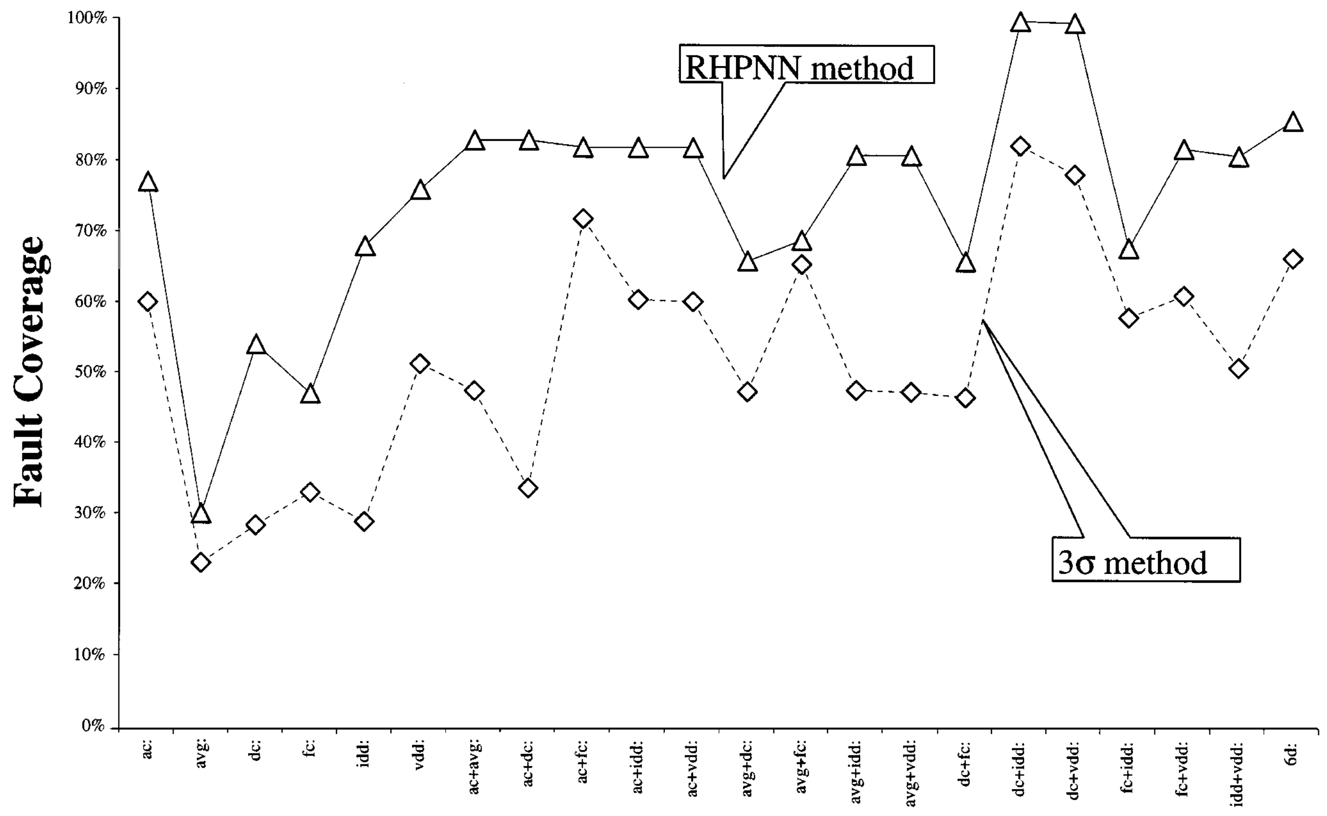

Fig. 5. Comparison between RHPNN and 3σ method for PLL.

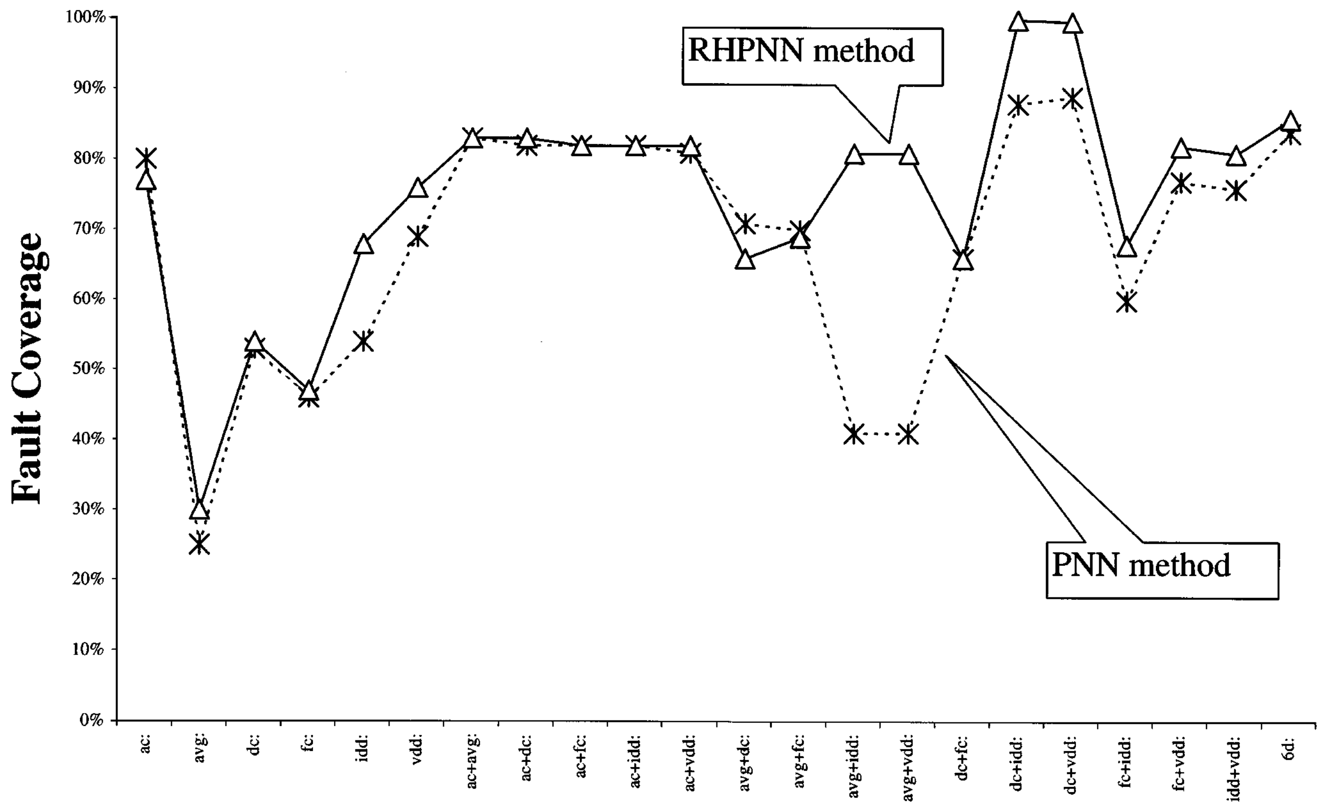

Fig. 6. Comparison between RHPNN and PNN for PLL.

TABLE VII
TRAINING AND TESTING TIME OF RHPNN

	Converging steps	Training time (s)	Testing time (s.10^{-4}/pattern)
AC voltage	5	7.55	1.69
DC voltage	14	20.59	1.54
AC current	44	62.57	1.96
DC current	5	7.53	1.89
AC	17	35.23	2.08
DC	41	75.49	2.16
Current	23	46.15	2.23
Voltage	26	51.31	2.08
All	20	59.21	6.16

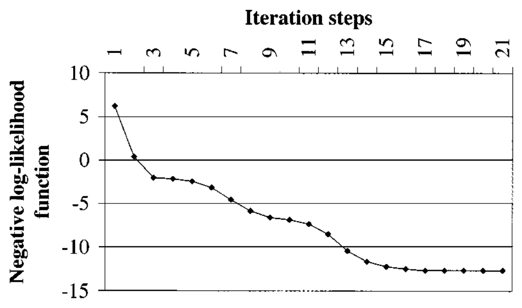

Fig. 7. Convergence of RHPNN training algorithm.

TABLE VIII
TRAINING AND TESTING TIME OF RHPNN, PNN, AND BPNN FOR 4-D MEASUREMENTS

	Converging steps	Training time (h:m:s)	Testing time / pattern (m:s)
BPNN	10,000	1:17:00	03:07
PNN	-	0:00:15	01:38
RHPNN	20	0:00:59	00:06

TABLE IX
RUN TIME BEHAVIOR OF APPLYING RHPNN

	Behavior
Converging steps	30
Training CPU time	20 minutes
Testing CPU time	53 seconds

V. CONCLUSION

The RHPNN is an extremely reliable technique for distinguishing good analog circuits from faulty. By examining the ac and dc voltage and current responses of an opamp stimulated with a single-frequency sinusoidal input, correct classification was obtained with an accuracy of 99.96%. Using other neural network techniques, the best result obtained was around 73%. For the second, more complex, example of a PLL, the RHPNN was in general more accurate than the simple threshold method or the basic PNN. Here, however, the choice of measurement parameters was significant choosing complementary parameters allowed correct classification to be obtained with an accuracy of 100%. Further the training and testing times for this technique were extremely small, making this approach ideally suited to production testing and test pattern generation.

REFERENCES

[1] Y. K. Malaiya and A. P. Jayasumana, "Enhancement of resolution in supply current based testing for large ICs," in *Proc. IEEE VLSI Test Symp.*, 1991, pp. 291–296.

[2] S. J. Spinks, C. D. Chalk, M. Zwolinski, and I. M. Bell, "Generation and verification of tests for analogue circuits subject to process parameter deviations," in *Proc. 1997 IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Paris, Oct. 1997, pp. 100–108.

[3] B. R. Epstein, M. Czigler, and S. R. Miller, "Fault detection and classification in linear integrated circuits: An application of discrimination analysis and hypothesis testing," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 102–113, Jan. 1993.

[4] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, pp. 178–188, 1936.

[5] Z. R. Yang and G. Musgrave, "Applying artificial neural network to fault diagnosis of analogue circuit," in *Proc. 3rd Int. Conf. VLSI and CAD*, Korea, Nov. 1993.

[6] S. Yu, B. W. Jervis, K. R. Eckersall, I. M. Bell, A. G. Hall, and G. E. Taylor, "Neural network approach to fault diagnosis in CMOS opamps with gate oxide short faults," *Electron. Lett.*, vol. 30, pp. 695–696, 1994.

[7] Y. Maidon, B. W. Jervis, and S. Lesage, "Diagnosis of multifaults in analogue circuits using multilayer perceptrons," *Proc. Inst. Elect. Eng. Circuits Devices Syst.*, vol. 144, pp. 149–154, 1997.

[8] S. S. Somayajula, E. Sanchez-Sinencio, and J. P. de Gyvez, "Analog fault diagnosis based on ramping power supply current signature clusters," *IEEE Trans. Circuits Syst.—II*, vol. 43, pp. 703–712, Oct. 1996.

[9] N. B. Hamida, K. Saab, D. Marche, B. Kaminska, and G. Quesnel, "LIMSoft: Automated tool for design and test integration of analog circuits," in *Proc. Int. Test Conf.*, 1993, pp. 571–580.

[10] M. Slamani and B. Kaminska, "Analog circuit fault diagnosis based on sensitivity computation and functional testing," *IEEE Design Test Comput.*, vol. 9, pp. 30–39, Jan. 1992.

[11] S. W. Director and R. A. Rohrer, "The generalized adjoint network and network sensitivities," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 318–323, Mar. 1969.

[12] Y. Chou, *Statistical Analysis*. New York: Holt Rinehart and Winston, 1970.

[13] T. Kohonen, *Self-Organization and Associative Memory, 3rd ed.*. Berlin, Germany: Springer-Verlag, 1989.

[14] R. L. Wilson and S. Ramesh, "Bankruptcy prediction using neural networks," *Decision Support Syst.*, vol. 11, pp. 545–557, 1994.

[15] Z. R. Yang, "UK construction company failure prediction: A robust heteroscedastic parzen window classifier," Ph.D., Univ. Portsmouth, Portsmouth, U.K., 1998.

[16] H. Ogawa and K. Yamasaki, "A theory of over-learning," *Artif. Neural Networks*, vol. 2, pp. 215–218, 1992.

[17] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995.

[18] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, Feb..

[19] D. Specht, "Probabilistic neural networks for classification, mapping, or associative memory," in *Proc. Int. Conf. Neural Networks*, vol. 1, 1988, pp. 525–530.

[20] ——, "Probabilistic neural networks," *Neural Networks*, vol. 3, pp. 109–118, 1990.

[21] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statistics*, vol. 3, pp. 1065–1076, 1962.

[22] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statistic Soc. (B)*, pp. 1–38, 1977.

[23] R. L. Streit and T. E. Luginbuhl, "Maximum likelihood training of probabilistic neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 764–783, Sept. 1994.

[24] Z. R. Yang and S. Chen, "Robust maximum likelihood training of probabilistic neural networks," *Neural Networks*, vol. 11, no. 4, pp. 739–747, 1998.

[25] R. G. Miller, "The jackknife—A review," *Biometrika*, vol. 61, pp. 1–15, 1974.

[26] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Exploration in the Cognition*. Cambridge, MA: MIT Press, 1986.

[27] W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Trans. Computer*, vol. C-18, pp. 401–409, May 1969.

[28] J. A. Leonard and M. A. Kramer, "Radial basis function networks for classifying process faults," *IEEE Contr. Syst. Technol.*, pp. 31–38, Apr. 1991.

[29] J. A. Leonard, M. A. Kramer, and L. H. Ungar, “Using radial basis functions to approximate a function and its error bounds,” *IEEE Trans. Neural Networks*, vol. 5, pp. 624–627, Apr. 1992.

Zheng Rong Yang received the B.S degree in electronic systems from Shanghai University, Shanghai, China, in 1982, and the Ph.D. degree from Portsmouth University, Portsmouth, U.K. in 1998.

He is currently a Research Associate in the Department of Physics at Heriot-Watt University, Edinburgh, Scotland. His research interests include numeric analysis, neural networks, evolution computation, statistical pattern recognition, financial forecasting, and medical data analysis.

Mark Zwolinski (M’92) received the B.Sc. and Ph.D. degrees in electronics from the University of Southampton, Southampton, U.K., in 1982 and 1986, respectively.

He is a Senior Lecturer in the Department of Electronics and Computer Science, University of Southampton. His research interests include simulation and modeling algorithms for analogue and mixed-signal integrated circuits, testing techniques for mixed-signal integrated circuits, and VHDL. He has co-authored over 50 research papers in technical journals and conferences.

Chris D. Chalk received the B.Sc. degree in physics with electronics from the University of East Anglia, East Anglia, U.K., in 1986. He received the M.Sc and Ph.D. degress from the University of Southampton, Southampton, U.K., in 1994 and 1998, respectively.

From 1986 to 1993, he was with by the British Broadcasting Corporation as a Radio Engineer. From 1994 to 1998, he was with Southampton University as a Research Assistant, working on mixed signal testing. He is currently with Philips Semiconductors, Southampton, as an Analog Designer. His interests include mixed-signal design and test, fault simulation, and automatic test stimulus generation for analog circuits.

Alan Christopher Williams (M’99) received the M.Eng. and Ph.D. degrees in electronic engineering from the University of Southampton, Southampton, U.K., in 1992, and 1997, respectively. His research focused on the development of high-level digital synthesis tools including VHDL and low-power design techniques.

He is currently a Research Lecturer in the Electronic Systems Design Group in the Department of Electronics and Computer Science, University of Southampton.

His current research interest focus on many aspects of high-level synthesis and simulation, including synthesis from system-level specifications, processor synthesis and the development of self-timed/asynchronous systems. He is also active in the development of low-cost FPGA and ASIC development tools for small to medium-scale industrial applications.