Analog Neural Network Design for RF Built-In Self-Test

Dzmitry Maliuk*, Haralampos-G. Stratigopoulos‡, He Huang* and Yiorgos Makris*
*Electrical Engineering Department, Yale University, 10 Hillhouse Ave, New Haven, CT 06520-8267, USA
‡TIMA Laboratory (CNRS-Grenoble INP-UJF), 46 Av. Félix Viallet, 38031 Grenoble, France

***Abstract*—A stand-alone built-in self-test architecture mainly consists of three components: a stimulus generator, measurement acquisition sensors, and a measurement processing mechanism to draw out a straightforward Go/No-Go test decision. In this paper, we discuss the design of a neural network circuit to perform the measurement processing step. In essence, the neural network implements a non-linear classifier which can be trained to map directly sensor-based measurements to the Go/No-Go test decision. The neural network is fabricated as a single chip and is put to the test to recognize faulty from functional RF LNA instances. Its decision is based on the readings of two amplitude detectors that are connected to the input and output ports of the RF LNA. We discuss the learning strategy and the generation of information-rich training sets. It is shown that the hardware neural network has comparable learning capabilities with its software counterpart.**

I. Introduction

The seemingly ever-increasing data rates and complexity of RF devices intensify the test effort that must be spent to guarantee the correct functioning of each manufactured part. Hundreds of sequential tests are carried out in practice to ensure the compliance with the specifications. This procedure demands sophisticated test equipment and results in long test times, which escalate significantly the overall manufacturing cost. As RF devices have become a ubiquitous part of our everyday lives, there is a great incentive to reduce the implicated test costs, while maintaining the highest possible quality and reliability of the parts that pass the test.

In the past decade, there has been an enhanced focus on alternative test strategies with the aim to simplify and standardize the test equipment and to increase the throughput, i.e. the number of parts tested in a given unit of time. These alterative test strategies can be broadly categorized as structural test, built-off test (BOT), built-in test (BIT), and implicit test. Structural tests aim to generate signatures that discriminate between faulty and functional circuits [1], [2]. They are devised specifically to detect a list of faults that result from a prior inductive fault analysis. This approach presents a high potential to reduce test costs, but further research is required on fault modeling. BOT relies on relatively simple circuits to convert RF signals to DC signals [3], [4]. These circuits are placed on the device load-board, thus allowing the interface of the device under test (DUT) to an inexpensive tester. BIT is a more aggressive approach. It consists of building a miniature tester on-chip [5], [6], reconfiguring the DUT in an easily testable form [7], and extracting on-chip informative measurements based on sensors [8], [9], [10], [11], [12], [13]. Implicit test is a generic technique which aims to infer the outcome of the standard tests based on low-cost measurements. This inference relies on statistical models that are built off-line using a representative training set of instances of the DUT. Different types of statistical models can be used, namely regression functions, in which case we can predict individually the high-level performances (this approach is known as alternate test [14], [15], [16], [17]), or classifiers, in which case we can make directly a Go/No-Go test decision [18].

In this paper, we explore the possibility of integrating a hardware version of a non-linear classifier along with the DUT, in order to execute a Go/No-Go BIT. The task of the classifier is to map on-chip measurements to a one-bit output, which simply indicates whether the DUT complies to its specifications or not. Furthermore, a true built-in self-test (BIST) is possible if the test stimuli are also generated on-chip. In particular, the classifier can replace the off-chip extraction and post-processing of measurements since it compacts the measurements to a binary Go/No-Go test response. Such a stand-alone BIST can then be performed on-line in the field of operation, in order to detect malfunctions due to environmental impact and wear. Thereby, it is of vital importance for devices that are deployed in safety-critical applications (e.g. avionics, medicine, nuclear reactors), sensitive environments (e.g. space operations), and remote-controlled systems.

To this end, we present the design of an analog neural network classifier as a single chip. The chip has been fabricated using a 0.5 μm digital CMOS process available through MOSIS and it is put to the test to learn to classify RF low noise amplifier (LNA) instances based on a BIT measurement pattern. This pattern is obtained by exercising the RF LNA with two single-tone sinusoidal stimuli of different powers and by recording the outputs of two amplitude (or peak or power) detectors that are placed at the input and output ports of the RF LNA [12]. The data sets that are used to train the chip and to validate its generalization capability on previously unseen RF LNA instances are generated using the technique in [19]. This technique enhances a real data set with the aim to generate synthetic data that populate the area around the classification boundary and, thereby, to improve training. In addition, with this technique we can generate an arbitrarily large validation set and, thereby, we can assess the classification rate with parts per million (PPM) accuracy. We demonstrate that the

978-1-4244-7207-9/10/$26.00 ©2010 IEEE

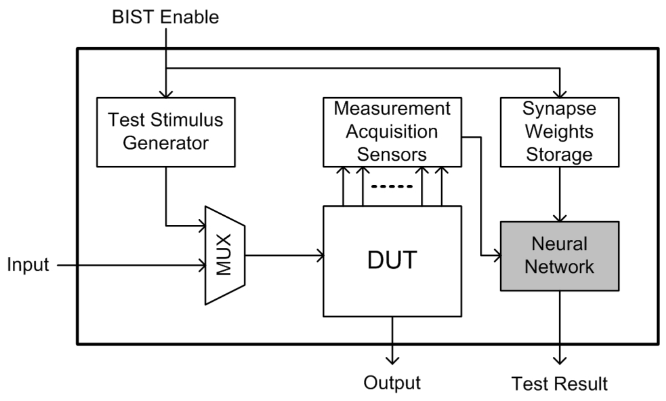

Fig. 1. BIST architecture.

hardware classifier achieves similar classification rates to its ideal software counterpart.

The paper is structured as follows. The following section discusses the complete BIST architecture and clarifies the focus of our work in this context. In section III, we discuss the design of the neural network classifier, its learning strategy, and we provide details of the fabricated chip. In section IV, we present the case study, namely the DUT and the corresponding BIT measurement pattern. In section V, we explain the procedure to generate the training and validation sets. In section VI, we put the chip to the test to classify BIT measurement patterns and we demonstrate its effectiveness compared to an ideal software classifier. In section VII, we discuss limitations and we point to future work. Finally, section VIII concludes the paper.

II. BIST Architecture and Overview

The BIST architecture is illustrated in Fig. 1. In an off-line training phase, the neural network classifier learns to map a measurement pattern to an one-bit output, which indicates whether this measurement pattern is a valid or invalid codeword, that is, whether the DUT complies to its specifications or not. Training is carried out on a sample set of fabricated chips, which is enhanced if necessary using the technique in [19]. The training phase results in an appropriate topology for the neural network and it also determines the weights of the internal synapses which are stored in a local memory. During the test phase, the weights are downloaded to the neural network. Next, the DUT is connected to the stimulus generator which enables a self-excitation of the DUT. The on-chip sensors monitor the DUT and provide the measurement pattern which is presented to the neural network. The neural network classifies the DUT by processing the measurement pattern and examining its footprint with respect to the learned classification boundary.

This approach presents a number of challenges. More specifically, the peripheral circuits that are dedicated to test should (a) incur low area overhead to minimize the extra die size cost (this would also imply a lower probability of fault occurrence within the test circuitry), (b) be non-intrusive, i.e they should have minimal interference with the DUT, (c) be more insensitive to process, voltage, and temperature (PVT) variations than the DUT itself, (d) make a prudent use of external pins, and (e) consume low power (this is crucial only for concurrent test or when test needs to be performed in the field in frequent idle times). Assuming that the above objectives are met, the overall success of this approach depends on the separation of the footprints of faulty and functional circuits when they are projected in the space that is formed by the selected measurements. If the faulty and functional classes are separable, the classifier will provide an accurate test decision. However, if the measurements do not perfectly discriminate between the two classes, that is, if the two classes overlap to some extent, then the classifier will inevitably misclassify the DUTs whose footprint falls in the ambiguous area. In effect, the misclassification rate is proportional to the volume of the ambiguous area. To reduce misclassification, the ambiguous area can be guard-banded with the option to retest the DUTs that lie in it [18].

The scope of this paper is to demonstrate a hardware classifier that can learn an optimal non-linear separation boundary given a measurement pattern. In other words, herein, we do not discuss the stimulus generation and the identification of discriminative measurements (which are both DUT-specific problems). Instead, we consider an LNA and a specifically related to it BIT measurement pattern as our case study. Our primary objective was to design a reconfigurable and flexible hardware classifier so as to study classification boundaries of various orders and gain insight into their performance and limitations compared to their software counterparts.

III. Neural Network Design

A. Background

Neural networks have an appealing silicon implementation. Synapses and computational elements can be densely interconnected to achieve high parallel distributed processing ability, which enables them to successfully solve complex cognitive tasks. Neural networks also provide a high degree of robustness and fault tolerance since they comprise numerous nodes that are locally connected, distributing knowledge among the numerous synapses. Thus, intuitively, damage to a few nodes does not impair performance. We are interested primarily in analog implementations of neural networks as, in comparison to a digital implementation, they have superior time response and computational density in terms of silicon mm^2 per operations per second and, in addition, they consume extremely low power.

In designing an analog neural network one has to consider a number of important factors. Appropriate connectionist topologies, training algorithms, long-term weight storage are among the most crucial. Furthermore, one has to consider implications of the technology in which a network is to be implemented. Digital CMOS processes, which are becoming more popular for analog/RF circuits, are plagued by process variation, mismatch, noise, environmental factors, etc. There is a large body of literature discussing these factors and various architectural considerations [20], [21], [22], [23]. The network

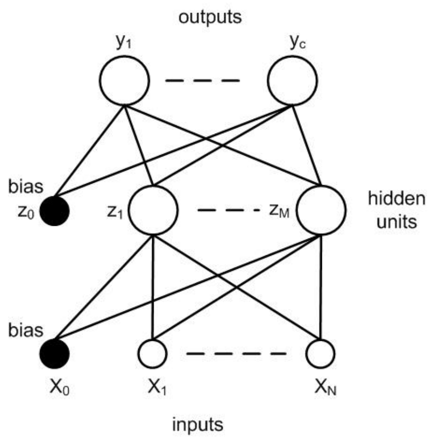

Fig. 2. 2-layer network diagram.

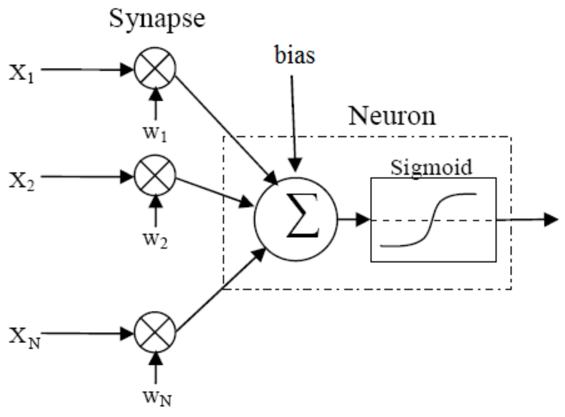

Fig. 3. Neuron and synapse models.

presented herein was designed with two key objectives in mind: reconfigurability, to explore various network models for a given application, and simplicity of training. The first was achieved by utilizing control bits to program the connectivity of the network, while the second by using digital weight storage in local RAM cells.

B. System Architecture

The chosen model for the neural classifier is a 2-layer network, as shown in Fig. 2. The network is feed-forward since it does not contain feedback loops, that is, each layer receives connections only from inputs or previous layers. The first layer has M units which receive the inputs $X_1, \cdots, X_N$ (e.g. the sensor-based measurements) and a constant $X_0 = 1$. The second layer has c units which receive the outputs of the first layer $z_1, \cdots, z_M$ and a constant $z_0 = 1$. The units in the first layer are called hidden units since they are not treated as output units. Similarly, the first layer is called hidden layer. Each unit in this network is a linear perceptron which has a simple mathematical model, as shown in Fig. 3. A synapse can be considered as a multiplier of an input signal value by the stored weight value. A neuron sums the output values of the connected synapses and passes the resulting sum through a nonlinear sigmoid activation function

$$g(\alpha) = \frac{1}{1 + \mathrm{e}^{-\alpha}}. \quad (1)$$

An explicit expression for the complete function represented by the diagram of Fig. 2 is

$$y_k = g\left(\sum_{i=0}^{M} w_{ki}^{(2)} z_i\right) \quad (2)$$

$$z_i = g\left(\sum_{j=0}^{N} w_{ij}^{(1)} X_j\right), \quad (3)$$

where $w_{ij}^{(k)}$ denotes the weight of input j for unit i in layer k and $w_{i0}^{(k)}$ denotes the bias for unit i in layer k. Such a network with 2 layers is capable of approximating any continuous functional mapping and can separate an arbitrary dichotomy (e.g. a given set of data points which have been labeled as belonging to one of two classes).

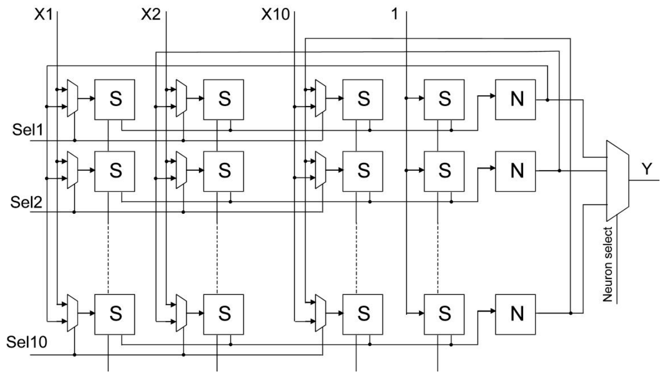

Fig. 4. Reconfigurable network architecture.

Fig. 4 illustrates the block-level schematic of a circuit implementation of a 2-layer network that can be reconfigured into any one-hidden-layer topology within the given number of inputs and neurons. The circuit consists of a matrix of synaptic blocks (S) and neurons (N). The synapses represent mixed-signal devices, in the sense that they conduct all computations in analog form while their weights are implemented as digital words stored in a local RAM memory. Multiplexers before each synapse are used to program the source of its input: either the primary input (for the hidden layer) or the output of a hidden unit (for the output layer). The results of synapse multiplication are summed and fed to the corresponding neuron, which performs a squashing function and produces an output either to the next layer or the primary output. The architecture is very modular and can easily be expanded to any number of neurons and inputs within the available silicon area. Therefore, the efficient implementation of the synapse and neuron circuits is essential for large networks. The output multiplexer is introduced to reduce the number of pins and ADCs. The signal encoding takes different forms: the outputs of the neurons are voltages, while the outputs of the synapses are currents. In

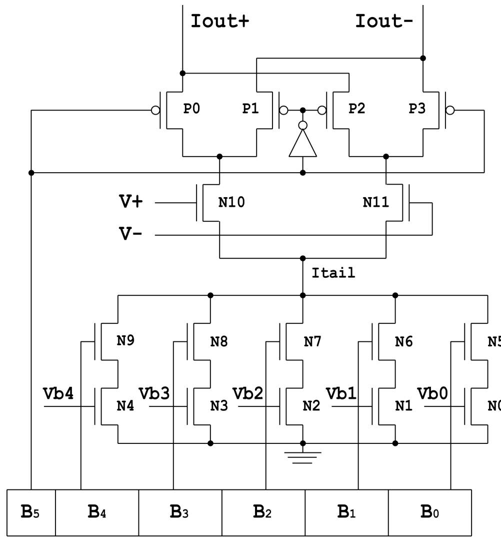

Fig. 5. Synapse circuit schematic.

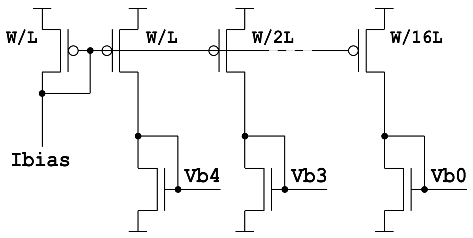

Fig. 6. Current sources control circuit.

addition, all signals are in differential form, thereby increasing the input range and improving noise resilience.

C. The Synapse Circuit

The basic function of a synapse is multiplication. Linear multiplication, as dictated by the mathematical model, is area expensive when implemented with analog circuitry. As a result, it is often the case that much simpler circuits exhibiting only approximate multiplication behavior are preferred. The effect of nonlinearity in synapse circuits can be addressed in various ways, with the solutions ranging from specially tailored backpropagation algorithms [20] to training algorithms independent of synapse and neuron transfer characteristics [24].

The synapse circuit chosen for this design is a simple multiplying DAC [25], which represents a differential pair with programmable tail current, as shown in Fig. 5. A differential input voltage is multiplied by the tail current producing a differential output current which is collected on the summing nodes common to all synapses connected to each neuron. The core of the circuit is a differential pair $N10 - N11$ performing a two-quadrant multiplication, while the four switching transistors $P0 - P3$ controlled by bit $B5$ steer the current between the two summing nodes, thus defining the sign of the multiplication. The tail current is digitally controlled by the five switch transistors $N5 - N9$ connecting the corresponding binary weighted current sources $N0 - N4$ to the tail node. Thus, the tail current as a function of a digital weight word (bits $B4 - B0$) can be represented by

$$I_{tail} = \sum_{i=0}^{4} B_i \cdot I_i = \sum_{i=0}^{4} B_i \cdot I_{bias} \cdot 2^{i-4} = I_{bias} \cdot W, \quad (4)$$

where B_i are the bits of a weight word, I_i is the current corresponding to the i-th bit, I_{bias} is the external biasing current, and W is the weight value. The biasing voltages Vb_i for the current sources of all synapses are supplied by a single biasing circuit shown in Fig. 6. The external biasing current I_{bias} sets the MSB current component, while the other currents are generated internally using the ratioed current mirrors.

The functional relation between input variables of the synapse circuit depends on the region of operation of the differential pair. For the saturation region, using the first order transistor models, this relation takes the following form

$$\Delta I_{out} = K_N \Delta V_{in} \sqrt{\frac{I_{bias} W}{K_N} - (\Delta V_{in})^2}, \quad (5)$$

where ΔV_{in} is the differential input voltage and K_N is the transconductance coefficient. Linear multiplication is only valid for a narrow range of differential input voltages. As can be seen from the above equation, this range depends on both the tail current and the transconductance coefficient. Since synapses account for the major part of power consumption, the tail current was kept minimum, yet enough for the differential pair to operate in saturation. The external biasing current during actual experimentation was about 1.5 μA producing the LSB current of about 90 nA, which corresponds to the weak inversion. In view of the small operating currents, the relatively wide input range (about 800 mV for maximum weight) was achieved by selecting input transistors with long channels (14 μm). A proper choice of the common mode input voltage alleviates the problem of the large gate-to-source voltage drops at the input transistors and keeps the current sources in saturation. The rest of the transistors were kept at minimum size.

D. The Neuron Circuit

The main function of a neuron circuit is to convert the sum of differential currents from its synapses into a differential voltage. Two issues need to be taken into account when designing this circuit. First, if the output voltage is propagated to the next layer, it should be compatible with the input requirements of the synapses, i.e. it should have high common mode. Second, the circuit should handle relatively large dynamic range of input currents. While the useful information is contained in the difference, the common mode current may vary significantly depending on the number of connected synapses, as well as on their weight values. In our design, the common mode current ranges from 90 nA (one synapse, minimum weight value) to 30 μA (10 synapses, maximum weight values).

A circuit satisfying these requirements is shown in Fig. 7. The central part of the circuit is responsible for common mode

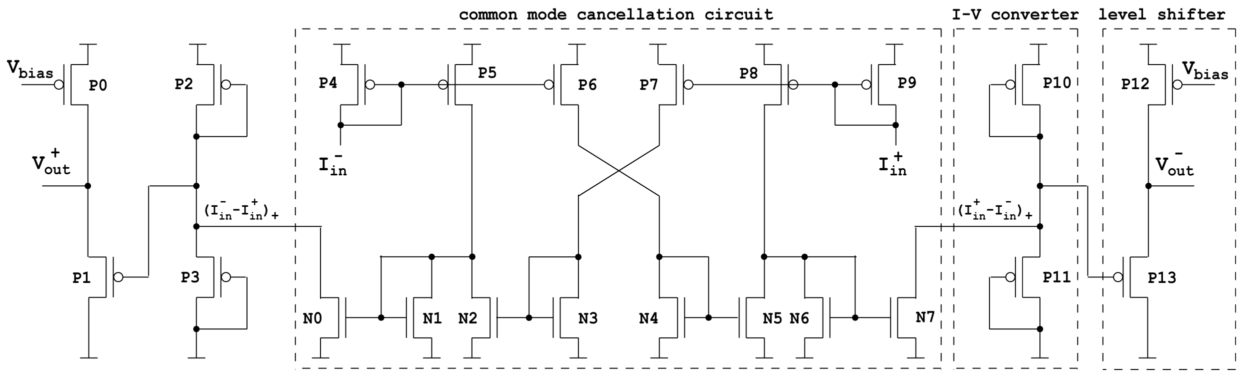

Fig. 7. Neuron circuit schematic.

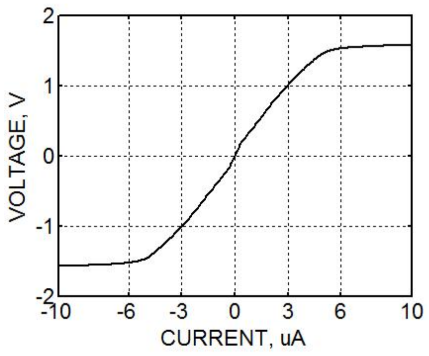

Fig. 8. Neuron sigmoid transfer function.

cancellation by subtracting the input currents from each other and producing a positive difference. The output currents of the transistors $N0$ and $N7$ can be expressed as $\max(0,(I_{in}^{-}-I_{in}^{+}))$ and $\max(0,(I_{in}^{+}-I_{in}^{-}))$, respectively. Thus, only one of the transistors can sink non-zero current at a time. The second stage is a simple current-to-voltage converter composed of two p-channel MOSFETs. It can be shown that, when the transistors are identical, such circuit exhibits a linear to the first degree characteristic of the following form

$$V = V_{dd} - \frac{I}{2K_P(V_{dd}-2V_{TP})}, \tag{6}$$

where K_P is the transconductance coefficient, V_{TP} is the threshold voltage, and V_{dd} is the supply voltage. The circuit also provides a limiting function when the input current exceeds the internal current flowing through the circuit, thus introducing nonlinearity to the neuron characteristic. Notice from the formula above that the slope of the characteristic depends on the K_P, which is set at the design stage by specifying transistor sizes. Finally, the output of the converter is shifted upwards to meet the requirements of the high common mode input voltage for the synapses in the following layer. This level shifter is a simple source follower circuit where the amount of shift is controlled by V_{bias}. A shift of 1V is used in this design. Fig. 8 shows the simulated transfer characteristic of the entire circuit and represents the activation function of the neuron.

E. Training Setup

When dealing with analog implementations the choice of training algorithms is generally limited. While on-chip implementations of backpropagation training exist, they suffer from the problems of limited precision and large area overhead. The practical scope is usually limited to either perturbation-based or optimized gradient-descent-based algorithms. In this work we pursue the former approach. It is particularly attractive since it requires little or no on-chip support resulting in a compact solution. The chip is trained by a computer program which uses the network's response to guide the search in the weight space. The gradient of the error surface cannot be calculated, since the model of the network is unknown in addition to the weights being non-continuous. Instead, we can probe different directions in the weight space by perturbing the weights and observing a change in the error function. Such weight perturbation-based algorithms are well suited for analog implementations, because they make no assumption about the network model, i.e. the synapse and neuron characteristics.

For this design we employed a popular algorithm called parallel stochastic weight perturbation [24]. In this algorithm, all weights are perturbed simultaneously by a random vector. Then the mean squared error is evaluated on the entire training set. If the error decreases, the new vector of weights is accepted; otherwise, it is discarded. This algorithm, however, suffers from high likelihood of convergence to a local minimum. Thus, training may need to be performed several times before a good solution is found. To decrease the probability of being stuck in a local minimum, this algorithm has been augmented with the simulated annealing technique, which is known to be efficient in avoiding local minima since it allows the state of the network to move "uphill". The main difference from the original algorithm consists in its ability to accept weight changes resulting in an increase of the error, however, with a certain probability. This probability depends on the magnitude of the error change and the "temperature" of the system T, i.e. $p \simeq \exp(-\Delta E/T)$. Higher temperatures at the initial stages favor the exploration of the whole search space. A cooling schedule is used to adjust the temperature and magnitude of weight perturbations as the training progresses.

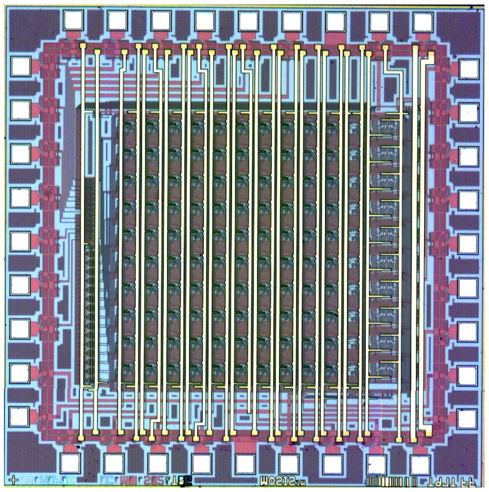
Fig. 9. Chip photograph.

TABLE I
CHIP KEY FEATURES.

Implementation method	mixed-signal
Network topology	reconfigurable 2-layer
IC process	0.5 μm CMOS
Core area	$1 \times 1.2\ mm^2$
Neurons	10
Synapses	100
Weight resolution	6 bit
Response time	$<\ 1\ ms$
Power supply	3.3 V
Max. current per synapse	3 μA

As will be shown later, this training approach applied to the hardware network has shown very similar performance in comparison to software neural networks.

F. Chip Fabrication

The chip has been fabricated using a 0.5 μm digital CMOS process available through MOSIS. Fig. 9 shows a photograph of the chip and Table I summarizes its key features. The training algorithm runs in MATLAB and communicates with the chip via a layer of two PCB boards, as shown in Fig. 10. The top layer is a custom-built PCB board that houses the chip, as well as two DACs, an ADC, and biasing circuits. The DACs produce differential input voltages to the network inputs, while the ADC is used to record the network response. The bottom layer is a commercial FPGA board that was used for communication with the PC and for programming the on-chip memory, the DACs, and the ADC.

IV. CASE STUDY

For our experiment, we selected the DUT to be a standard source-degenerated LNA. Its topology is shown in Fig. 11. The initial design targeted the set of nominal performances listed in Table II. The LNA is integrated along with two RF amplitude detectors that are placed at its input and output ports. The amplitude detectors provide DC signals proportional to the RF power seen at their inputs. These DC signals comprise the BIT measurement pattern that is finally presented to the classifier. The objective is to have very simple sensor structures with small area and power overhead, such that they qualify for a BIST application. The schematic of the selected amplitude detector is presented in Fig. 12 and its key features are summarized in Table III. The circuits were designed in the TSMC 0.18 μm CMOS process and were integrated into a single die as shown in Fig. 13.

A set of 1000 instances reflecting manufacturing process variations is generated through a Monte Carlo post-layout simulation. The parameter distribution and mismatch statistics were provided by the design kit. The amplitude detectors undergo the same process variations as the LNA. Next, each instance is fully characterized using standard test benches to

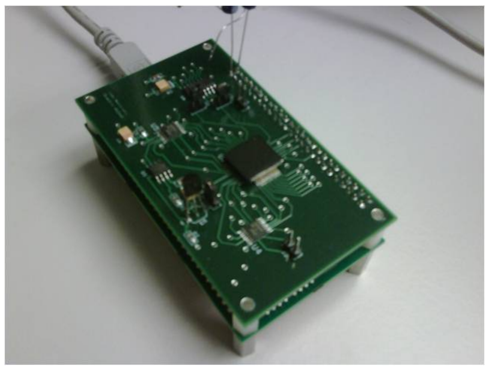
Fig. 10. Test board.

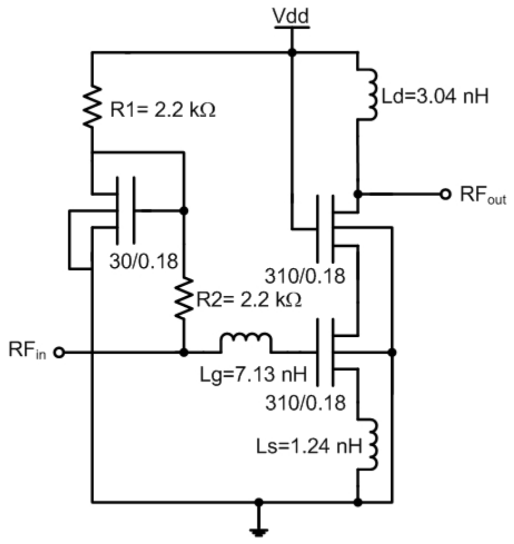

Fig. 11. LNA schematic.

TABLE II
LNA FEATURES.

Performance	Nominal
Central frequency (in GHz)	1.575
NF (in dB)	2
S11 (in dB)	-10
Gain (in dB)	15
IIP3 (in dBm)	-5
S22 (in dB)	-10
Power consumption (in mW)	18

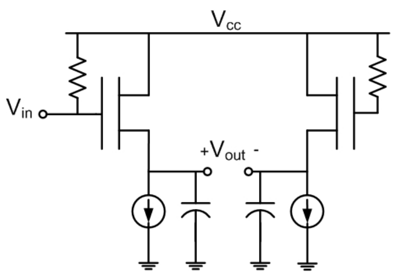

Fig. 12. Differential topology amplitude detector.

TABLE III
POWER DETECTOR FEATURES.

Frequency band (in GHz)	1.575
Dynamic Range (in dB)	50
Power consumption (in μW)	6.66
Area overhead (in μm^2)	42 × 80

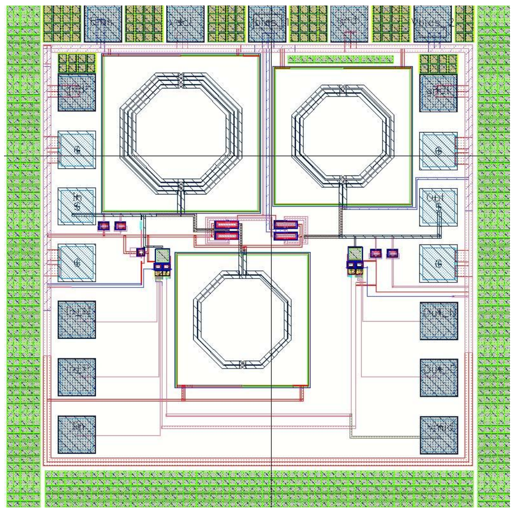

Fig. 13. LNA and amplitude detectors layout.

obtain the set of performance parameters. In addition, we obtain a four-dimensional BIT measurement pattern (e.g. the test signature) by applying to the LNA two test stimuli of powers -10 and 0 dBm consisting of a single tone at 1.575 GHz. It should be noted that the power levels of these test stimuli are much higher than that of typical signals expected during a real application. This is because the input amplitude detector does not have enough gain to detect small incoming signals, in response to which its output is likely to be hidden below the noise level.

For the original sample of fully characterized devices we set the specification limits to Mean $\pm$ 3·StdDev (where $+$ or $-$ depends on whether the performance has a lower or upper limit). The Mean and StdDev of individual performances are computed across the set of 1000 devices. Our choice of specifications resulted in 8 faulty devices out of 1000. Fig. 14 illustrates the distribution of nominal and faulty devices projected onto a 2-dimensional subspace of the measurement pattern. In particular, the horizontal and vertical axes correspond to the normalized input and output amplitude detector readings when excited by -10 dBm signal. The 8 faulty devices fall in the tail of the distribution which is sparsely populated. For training the classifier, however, it is preferable to have a balanced data set, such that one class does not overshadow the other. Furthermore, sparsely populated areas produce randomness in the curvature of the separation boundary and, thereby, can deteriorate the classification. Therefore, the training set must be populated with many marginal devices whose footprint lies in close proximity to the boundary. Similarly, we would like to evaluate the classification rate using a larger validation set. In the following section, we briefly discuss a technique to generate the desired training and validation sets.

V. DATA SET GENERATION

A Monte Carlo simulation samples with priority the statistically likely cases. Therefore, it is typically required to go through many passes so as to assemble a training set with an adequate number of marginal devices. Notice that a few simulations targeting only process and environmental corners do not suffice because they do not account for the actual statistics. To alleviate the computational effort, we adopt the technique in [19].

The main idea is to use the available set of 1000 devices to estimate the joint probability density function of the performances and the measurement pattern. Subsequently, the estimated density can be sampled to obtain new observations that correspond to synthetic devices following the original distribution. Sampling can be performed very fast, in particular 1 million observations can be obtained in 25 minutes.

To build the training set, we sample iteratively devices from the density with the aim to generate three equal sets, a set of faulty devices, i.e. devices with at least one performance falling outside the 3·StdDev specification limit, a set of marginally-in-the-specifications devices, i.e. devices that have at least one of their performances falling between the 2·StdDev and 3·StdDev specification limits, and a set of functional devices whose performances fall within the 2·StdDev specification limit. The generated training set is illustrated in Fig. 15, which shows a marked contrast to Fig. 14 that contains the original devices. The generated training set contains 900 devices, of which 1/3 are faulty, 1/3 are marginally functional, and 1/3 are functional close to the nominal design. Now the boundary can be better approximated since the area around it is populated with many samples.

Notice that this training set is "biased" in the sense that it is enhanced with a large number of faulty devices. In contrast, the trained classifier is validated using a large validation set of devices (e.g. 1 million) that is produced by "natural" sampling of the density. In this way, the validation set emulates a large set of devices that will be seen in production. As an example, Fig. 16 illustrates the original set of 1000 devices together with 10^4 randomly generated synthetic devices. This technique allows us to decompose the classification error into test escape (e.g. probability of a faulty device passing the test) and yield loss (e.g. probability of a functional device failing the test)

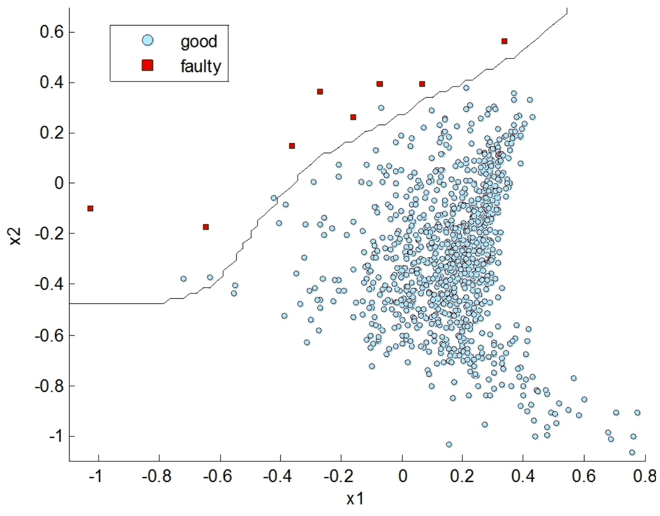

Fig. 14. Original data set consisting of 1000 devices generated through the Monte-Carlo simulation.

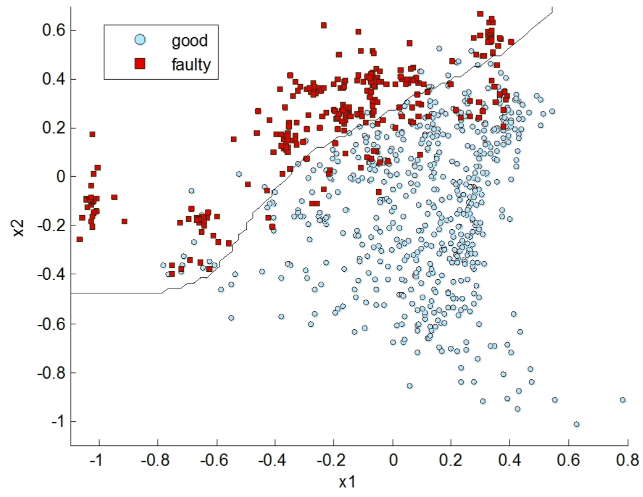

Fig. 15. Synthetic training set derived through resampling of the original distribution. It is equally represented by faulty, marginally functional, and functional devices. The decision boundary is generated by the hardware classifier.

and express these test metrics with PPM accuracy. Formally, let C_g and T_p denote the event that a device is functional and that a device passes the test, respectively, and let C_g^c and T_p^c denote the complementary events, i.e. a device is faulty and a device does not pass the test, respectively. Given a large sample of one million devices, test escape (T_E) and yield loss (Y_L) estimates can be derived using relative frequencies [26]:

$$T_E^* = \frac{N_{C_g^c \bigcap T_p}}{N_{T_p}} \tag{7}$$

$$Y_L^* = \frac{N_{C_g \bigcap T_p^c}}{N_{C_g}}, \tag{8}$$

where N_X denotes the count of event X.

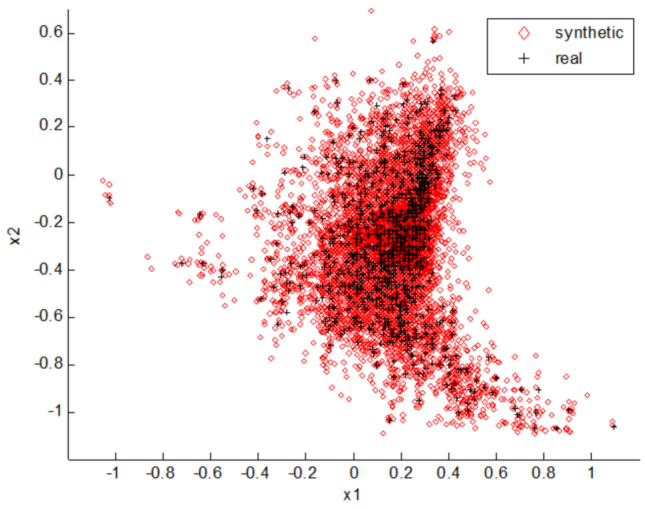

Fig. 16. Original and synthetic devices shown together. One million of synthetic devices comprise the validation set.

TABLE IV
CLASSIFIER PERFORMANCE.

Number of hidden neurons	2	4	8
Software network			
Training error, %	5.82	4.91	4.88
Validation error, %	0.566	0.548	0.581
Hardware network			
Training error, %	6.82	5.53	5.75
Validation error, %	0.727	0.435	0.491

VI. Experimental Results

We experiment with three different neural network configurations that consist of a single hidden layer with 2, 4, and 8 neurons, respectively. The training is repeated 5 times for each network configuration to average out the randomness due to stochastic nature of the training algorithm. For comparison purposes, the same experiments are repeated with software neural networks of identical topologies using the Matlab Neural Networks toolbox. The software training is carried out using the resilient backpropagation algorithm.

The results on the training and validation sets are presented in Table IV. A large discrepancy between the training and validation errors is the result of having a "biased" training set with many marginal devices and a "natural" validation set where the majority of devices is distributed around the nominal point. In terms of training error, the software classifier consistently outperforms the hardware classifier by about 1%. However, the validation errors, representing the true accuracy of classification, are similar for both networks (the difference is $< 0.2\%$). The hardware version achieves even smaller error for the models with more than 2 hidden neurons. In fact, the best performance is shown by the hardware network with 4 hidden neurons resulting in the error of 0.435% or equivalently 4350 misclassified PPM.

The validation error can be decomposed into T_E and Y_L. Since T_E is more undesirable than Y_L, the decision boundary can be biased to favor Y_L over T_E, i.e. to accept fewer faulty

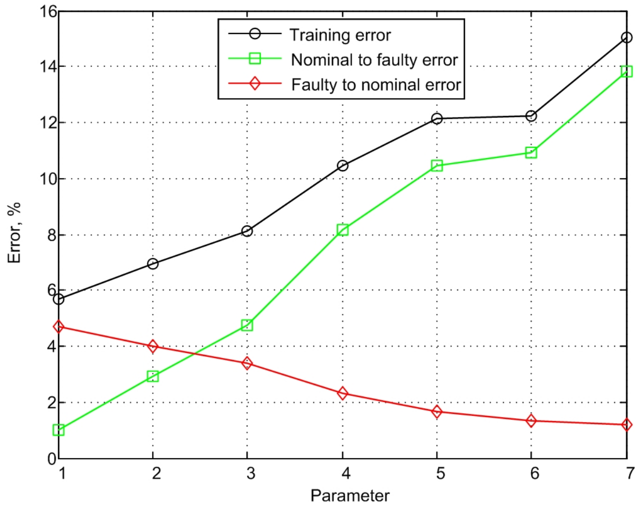

Fig. 17. Training error decomposed into false positives and false negatives for biased boundaries. Larger values of the parameter shift the boundary towards the functional class resulting in lower "faulty-to-nominal" error.

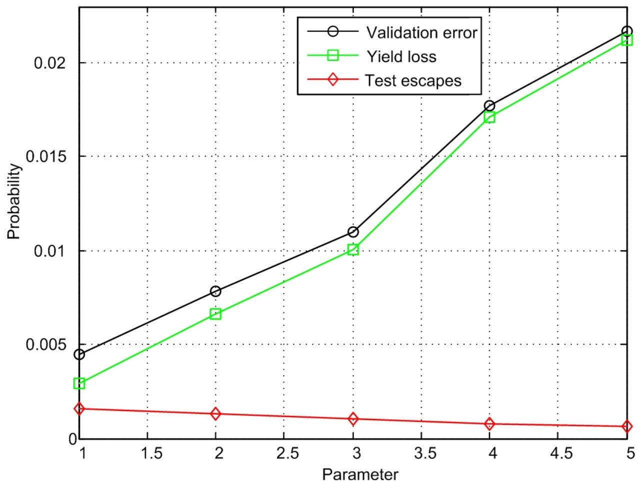

Fig. 18. Test escape and yield loss error for biased boundaries obtained on the validation set.

devices at the expense of rejecting more functional devices. To this end, the hardware network with 4 hidden neurons is repeatedly trained for different values of a parameter that regulates the positioning of the boundary. Larger values of the parameter correspond to higher penalties imposed on T_E and cause the boundary to shift towards the functional class. The results on the training set are illustrated in Fig. 17, which shows the reduction of misclassification of faulty devices as we gradually increase the value of the parameter (the value 1 corresponds to an "unbiased" boundary, see Table IV). Fig. 18 illustrates the trade-off between T_E and Y_L obtained on the validation set. When the parameter value equals 3 we obtain $Y_L \approx 10 \cdot T_E$. This trade-off point corresponds to the rule-of-ten, which suggests that it is 10 times more expensive to accept a faulty device than rejecting a functional one. The absolute numbers for T_E and Y_L are 1062 and 10063 PPM, respectively. While these numbers are rather large, we emphasize that the primary purpose of this work was to demonstrate the equivalence of the hardware classifier to its software counterpart. More discriminative measurements or a larger measurement pattern are needed to reduce this error to industry acceptable levels.

VII. Discussion

The learning capability of our hardware neural network was assessed in comparison to software neural networks of identical topologies. As is evident from the error on the training set, the hardware network falls short of the ideal software network in terms of its classification accuracy. We believe that this is due to its limited capacity resulting from the high nonlinearity in synapse multiplication, the limited resolution and dynamic range of weight values, and the limitations of the training algorithm. The dynamic range limitation, for instance, exhibited itself as saturation of weights to their maximum values, which was often observed during the experiments.

The problem of limited dynamic range can be addressed by designing synapses with adjustable gain and executing gain change whenever a weight becomes either too small or saturates. Alternatively, the dynamic range of all synapses connected to one neuron can be effectively adjusted by changing the gain of this neuron [27]. The problem of weight resolution is more subtle, especially in presence of high nonlinearity. The minimum number of bits required depends on the network architecture and is generally problem-specific. Comparing the training outcomes of the software network performed in double precision arithmetic with the hardware version suggests that 6 bits might be insufficient. However, increasing weight resolution with minimum size devices is limited by the inherent mismatch and parameter variation [28]. The problem is compounded by the fact that the effective resolution due to nonlinear weights/multiplication is inevitably lower. Finally, the training algorithm itself affects the performance of the classifier in terms of training time and its ability to find a global solution. In the course of experiments, the simulated annealing-based parallel weight perturbation algorithm demonstrated good convergence properties and low variance of the final error with the only drawback being lengthy training time. A more elaborate training will incorporate the gradient information leading to significant speedup.

The issue of weight storage is central to an analog neural network implementation. It is usually addressed by using local digital memory, storing weights as capacitor charge, or using non-volatile memory. The digital memory employed in this design is a fast and convenient way to explore training algorithms and network performance. However, in a BIST application, this would require an external memory to store the weights permanently. An attractive solution to non-volatile weight storage can be offered by floating gate transistors, which can be implemented in standard CMOS [29], [30]. Unlike digital FLASH memories, however, such transistors are primarily intended for storing analog values quantified by the amount of charge on their floating gates. In addition, they can be used as a part of the computational circuit leading to significant reduction in area overhead. Successful application of the floating gate technology will require resolving several issues, such as precision programming schemes, high voltage handling, unidirectional weight update, etc., yet this is the direction of choice for continuing this work.

Finally, a critical issue for the overall success of on-line BIST, though not directly related to the hardware implementation, is the generation of a representative training set. When obtained by post-production characterization of a number of chips, it faithfully reflects the manufacturing process statistics, thus it is well suited for production testing. However, it still has to be investigated whether this scheme adequately represents the effects of degradation during the lifetime of the device, and, if not, what corrections to the data generation process would be required.

VIII. Conclusions

In this paper, we discussed an analog hardware implementation of a neural network which can be employed to map sensor-based measurements to a compact Go/No-Go test decision. This mapping function is useful in the context of a stand-alone BIST for manufacturing testing or for in-field testing. We discussed in detail the design of the basic blocks of the neural network, namely the synapse and the neuron, as well as its learning strategy. We demonstrated its capacity to perform pattern recognition using a data set from LNA instances. Finally, we discussed its limitations compared to an identical software version and we pinpointed the improvements that can be made to arrive at a cost-effective implementation.

Acknowledgments

This research has been carried out with the support of the National Science Foundation (NSF ECS-0622081 and CCF-0702522), the Semiconductor Research Corporation (SRC-1836.029) and a Marie Curie International Reintegration Grant (MIRG-CT-2007-209653).

References

[1] E. Silva, J. Pineda de Gyvez, and G. Gronthoud, "Functional vs. multi-VDD testing of RF circuits," in *IEEE International Test Conference*, 2005, pp. 17.2.1–17.2.9.

[2] E. Acar and S. Ozev, "Defect-oriented testing of RF circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 5, pp. 920–931, 2008.

[3] J. Ferrario, R. Wolf, S. Moss, and M. Slamani, "A low-cost test solution for wireless phone RFICs," *IEEE Communications Magazine*, vol. 41, no. 9, pp. 82–88, 2003.

[4] S. Bhattacharya and A. Chatterjee, "A DFT approach for testing embedded systems using DC sensors," *IEEE Design & Test of Computers*, vol. 23, no. 6, pp. 464–475, 2006.

[5] M. M. Hafed, N. Abaskharoun, and G. W. Roberts, "A 4-GHz effective sample rate integrated test core for analog and mixed-signal circuits," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 4, pp. 499–514, 2002.

[6] J.-Y. Rye and B. C. Kim, "Low-cost testing of 5 GHz low noise amplifiers using new RF BIST circuit," *Journal of Electronic Testing: Theory and Applications*, vol. 21, no. 6, pp. 571–581, 2005.

[7] R. Ramzan, L. Zou, and J. Dabrowski, "LNA design for on-chip RF test," in *IEEE International Symposium on Circuits and Systems*, 2006, pp. 4236–4239.

[8] A. Gopalan, M. Margala, and P. R. Mukund, "A current based self-test methodology for RF front-end circuits," *Microelectronics Journal*, vol. 36, no. 12, pp. 1091–1102, 2005.

[9] M. Cimino, H. Lapuyade, M. De Matos, T. Taris, Y. Deval, and JB. Bégueret, "A robust 130nm-CMOS built-in current sensor dedicated to RF applications," in *IEEE European Test Symposium*, 2006, pp. 151–158.

[10] Y.-C. Huang, H.-H. Hsieh, and L.-H. Lu, "A low-noise amplifier with integrated current and power sensors for RF BIST applications," in *IEEE VLSI Test Symposium*, 2007, pp. 401–408.

[11] D. Mateo, J. Altet, and E. Aldrete-Vidrio, "Electrical characterization of analogue and RF integrated circuits by thermal measurements," *Microelectronics Journal*, vol. 38, no. 2, pp. 151–156, 2007.

[12] A. Valdes-Garcia, R. Venkatasubramanian, J. Silva-Martinez, and E. Sánchez-Sinencio, "A broadband CMOS amplitude detector for on-chip RF measurements," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 7, pp. 1470–1477, 2008.

[13] L. Abdallah, H.-G. Stratigopoulos, C. Kelma, and S. Mir, "Sensors for built-in alternate RF test," in *IEEE European Test Symposium*, 2010, pp. 49–54.

[14] S. S. Akbay, J. L. Torres, J. M. Rumer, A. Chatterjee, and J. Amtsfield, "Alternate test of RF front ends with IP constraints: Frequency domain test generation and validation," in *IEEE International Test Conference*, 2006, pp. 4.4.1–4.4.10.

[15] S. Ellouz, P. Gamand, C. Kelma, B. Vandewiele, and B. Allard, "Combining internal probing with artficial neural networks for optimal RFIC testing," in *IEEE International Test Conference*, 2006, pp. 4.3.1–4.3.9.

[16] R. Voorakaranam, S. S. Akbay, S. Bhattacharya, S. Cherubal, and A. Chatterjee, "Signature testing of analog and RF circuits: Algorithms and methodology," *IEEE Transactions on Circuits and Systems - I*, vol. 54, no. 5, pp. 1018–1031, 2007.

[17] H.-G. Stratigopoulos, S. Mir, E. Acar, and S. Ozev, "Defect filter for alternate RF test," in *IEEE European Test Symposium*, 2009, pp. 101–106.

[18] H.-G. Stratigopoulos and Y. Makris, "Error moderation in low-cost machine-learning-based analog/RF testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 2, pp. 339–351, 2008.

[19] H.-G. Stratigopoulos, S. Mir, and Y. Makris, "Enrichment of limited training sets in machine-learning-based Analog/RF testing," in *Design, Automation and Test in Europe Conference*, 2009, pp. 1668–1673.

[20] J. Lont and W. Guggenbuhl, "Analog CMOS implementation of a multilayer perceptron with nonlinear synapses," *IEEE Transactions on Neural Networks*, vol. 3, no. 3, pp. 457–465, 1992.

[21] M. Milev and M. Hristov, "Analog implementation of ANN with inherent quadratic nonlinearity of the synapses," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1187–1200, 2003.

[22] S. Tam and M. Holler, "An electrically trainable artificial neural network (ETANN) with 10240 floating gate synapses," in *Proceedings of International Joint Conference on Neural Networks*, 1989, pp. 191–196.

[23] A. Montalvo, "Toward a general-purpose analog VLSI neural network with on-chip learning," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 413–423, 1997.

[24] M. Jabri and B. Flower, "Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 154–157, 1992.

[25] V. F. Koosh and R. M. Goodmanr, "Analog VLSI neural network with digital perturbative learning," *IEEE Transactions on Circuits and Systems - II*, vol. 49, no. 5, pp. 359–368, 2002.

[26] H.-G. Stratigopoulos, S. Mir, and A. Bounceur, "Evaluation of analog/RF test measurements at the design stage," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 4, pp. 582–590, 2009.

[27] M. Hohfeld and S. E. Fahlman, "Probabilistic rounding in neural network learning with limited precision," in *Neurocomputing 4*, 1992, pp. 291–299.

[28] T.-S. Gotarredona B.-L. Barranco and R.-S. Gotarredona, "Compact low-power calibration mini-DAC for neural arrays with programmable weights," *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1207–1216, 2003.

[29] P. Hasler R. R. Harrison and B. A. Minch, "Floating gate CMOS analog memory cell array," in *IEEE International Symposium on Circuits and Systems*, 1998, pp. 202–207.

[30] P. Hasler, "Floating-gate devices, circuits, and systems," in 5^{th} *International Workshop on System-on-Chip for Real-Time Applications*, 2005, pp. 482–487.