Fault Classification in Phase-Locked Loops Using Back Propagation Neural Networks

Jayabalan Ramesh, Ponnusamy Thangapandian Vanathi, and Kandasamy Gunavathi

Phase-locked loops (PLLs) are among the most important mixed-signal building blocks of modern communication and control circuits, where they are used for frequency and phase synchronization, modulation, and demodulation as well as frequency synthesis. The growing popularity of PLLs has increased the need to test these devices during prototyping and production. The problem of distinguishing and classifying the responses of analog integrated circuits containing catastrophic faults has aroused recent interest. This is because most analog and mixed signal circuits are tested by their functionality, which is both time consuming and expensive. The problem is made more difficult when parametric variations are taken into account. Hence, statistical methods and techniques can be employed to automate fault classification. As a possible solution, we use the back propagation neural network (BPNN) to classify the faults in the designed charge-pump PLL. In order to classify the faults, the BPNN was trained with various training algorithms and their performance for the test structure was analyzed. The proposed method of fault classification gave fault coverage of 99.58%.

Keywords: Fault classification, back propagation neural network, PLL testing, charge-pump, phase frequency detector.

Manuscript received Feb. 28, 2008; revised June 21, 2008; accepted July 01, 2008.

Jayabalan Ramesh (Phone: +919894169253, email: jramesh60@yahoo.com), Ponnusamy Thangapandian Vanathi (ptvani@yahoo.com), and Kandasamy Gunavathi (kgunavathi2000@yahoo.com) are with the ECE Department, PSG College of Technology, Coimbatore, Tamil Nadu, India.

I. Introduction

The integrated circuit (IC) manufacturing costs are strongly affected by the cost of test equipment, test time, and test procedure development. This is especially true in mixed-signal ICs when analog blocks are involved. Furthermore, the cost for the analog part often dominates the total cost of testing, while the analog circuitry represents only a small percentage of the total area. In an attempt to minimize production costs, IC manufacturers are now investing considerable effort in the area of mixed analog/digital design and testing. The designed charge-pump phase-locked loops (CP-PLLs) employ a phase frequency detector (PFD) and a charge pump instead of a conventional phase detector (PD) and a low-pass filter (LPF) in the generic architecture. The combined PFD and charge-pump circuit offers several advantages.

- The capture range is only limited by the voltage-controlled oscillator (VCO) output frequency range.
- The static phase error is zero if mismatches and offsets are negligible.
- The PLL does not suffer from false lock.
- The input signal and the VCO output are exactly in phase.

All integrated circuits, fault-free or otherwise, are subject to parametric changes due to process variations [1]. Even though the circuits are designed to have some tolerance, some catastrophic faults can vary up to 3σ; therefore, a satisfactory threshold should be selected for fault classification. Since this threshold selection method was insufficient for fault classification [2], a statistical method can be selected. Linear discrimination analysis (LDA), a classical statistical method, was employed in 1993 for fault classification [2]. Limitations in LDA led Epstein to apply neural networks for fault classification [3]-[5], which gave better result when compared to other statistical methods. In our study presented this paper, the back propagation neural network (BPNN) is applied to fault classification in CP-PLL.

For all the transistors, the various structural (catastrophic) faults were modeled and extensively simulated using Tanner Tools Professional. The training and testing sets were then obtained from a Monte Carlo simulation of the faulty and fault free circuit results for the fault classification using BPNN.

II. Test Structure

The charge-pump PLL [6] shown in Fig. 1 is used to evaluate the BPNN method of fault classification.

The CP-PLL [7] consists of the following blocks.

- PFD: It generates pulses (UP and DOWN signals) as wide as the phase difference between *data* and *dclock*.
- Charge pump: It converts the logic states of the PFD into analog signals suitable for controlling the VCO.
- Loop filter: It minimizes noise in the signal with the maximum response time.
- VCO: It generates frequency based on input voltage from loop filter.
- Divide-by-*n* counter: It generates *dclock* signal based on VCO output frequency which is fed back to the PFD.

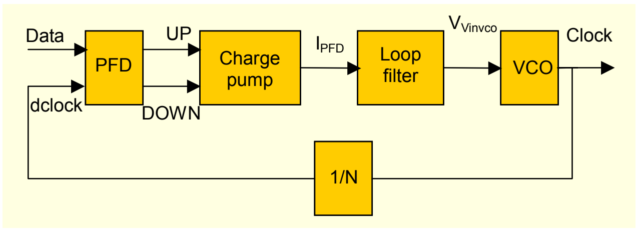

Fig. 1. Block diagram of the charge-pump PLL.

III. Phase Frequency Detector

PLLs incorporating sequential-logic, namely, PFDs have been widely used in recent years [8]. Reasons for their popularity include extended tracking range, frequency-aided acquisition, and low cost. The output of the PFD depends on both the phase and frequency of the inputs. The phase frequency detector shown in Fig. 2 employs sequential logic to create three states to respond to the rising and falling edges of A (connected to *data*) and B (connected to *dclock*). The outputs Q_A and Q_B are called UP and DOWN, respectively. The UP and DOWN signals can be false simultaneously or either one can be true, but both can never be true simultaneously. Therefore, the three states allowed by a PFD are termed as UP, DOWN and N, where N denotes neutral. The output of the PFD is shown in the Figs. 3 (a) and (b).

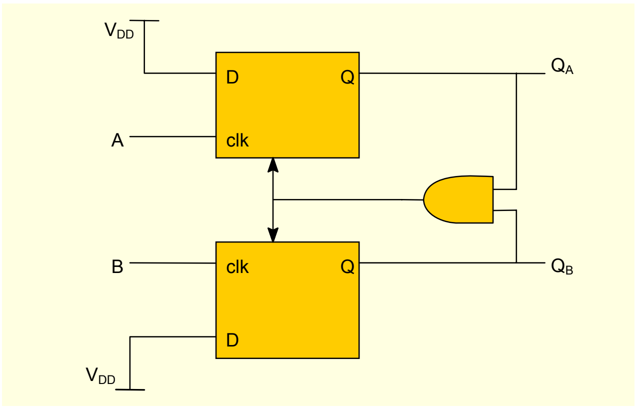

Fig. 2. PFD using a D-flip flop.

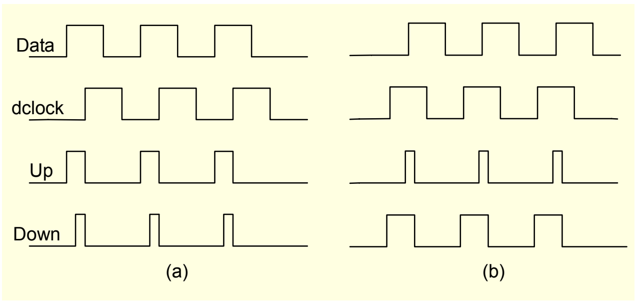

Fig. 3. (a) Up signal generation and (b) down signal generation.

IV. Charge Pump

The purpose of the charge pump is to convert the logic states of the PFD into analog signals suitable for controlling the VCO. The CMOS implementation of the charge-pump circuit with a cascode current mirror is shown in Fig. 4. The charging and discharging of this circuit operates at a faster rate than the push/pull current source charge pump. This kind of charge pump is used in high-speed applications. When the UP signal is 0, the PMOS transistor conducts and keeps the charge pump in charging mode. When the DOWN signal is 1, the NMOS transistor conducts and keeps the charge pump in discharge mode.

V. Voltage Controlled Oscillator

A VCO operates as a variable length, variable delay ring oscillator having a current-starved inverter and an anti-high-gain circuit for each stage as shown in Fig. 5. Broadband operation of this VCO can be achieved by cascading the different stages. The VCO uses a tuning voltage or control voltage, V_c, generated by the charge pump, to speed up or slow down the PLL's output frequency as the PLL attempts to lock.

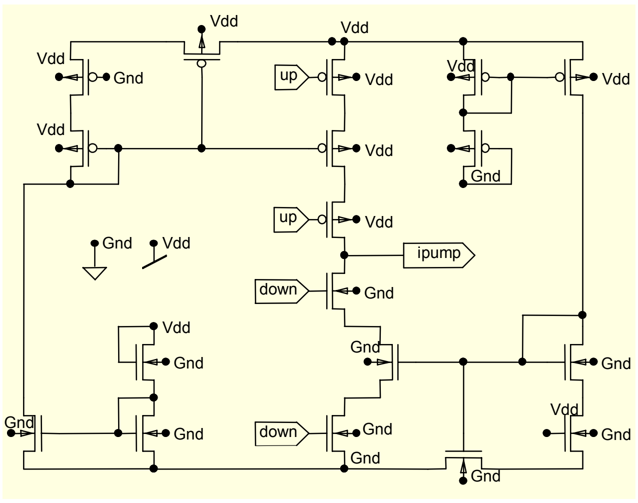
Fig. 4. T-spice schematic of the charge-pump circuit.

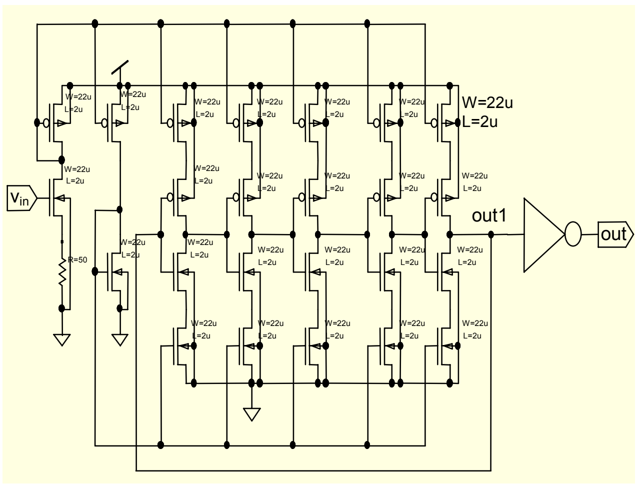
Fig. 5. T-spice implementation of current-starved VCO.

A ring oscillator is chosen due to the ease of design and implementation. Compared to LC oscillators, ring oscillators have a wide tuning range and small area consumption, but poor phase-noise performance. The ring consists of 5 stages of inverters to yield the desired frequency of operation from 15 MHz to 500 MHz, which is suitable for generating the clock signal of low to medium performance microcontrollers. The advantages of using a current-starved VCO are that it has a comparatively simpler design and leads to efficient area usage. As the VCO is starved of current, power efficiency is also improved.

VI. Divide-by-*n* Counters

The output of the VCO has to be divided before it is fed back to the input of the PLL. A programmable divider is used, which receives a reference clock signal of a predetermined frequency and is structured to divide the reference clock signal by *n* and provide an output pulse signal for every *n* cycles of the reference clock signal. The divide-by-2 counter is shown schematically in Fig. 6. It produces one pulse at the output for every two cycles.

Table 1 shows the design specifications of the PLL.

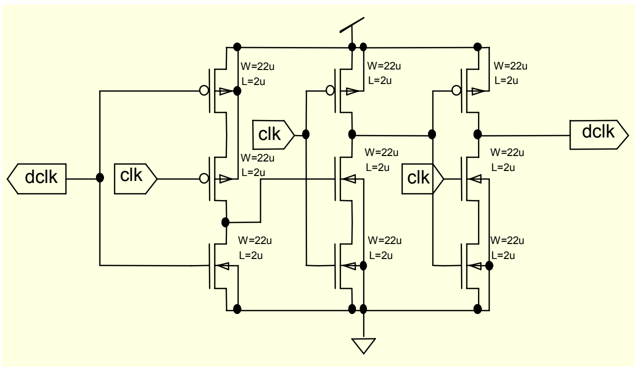
Fig. 6. T-spice schematic of the divide-by-2 counter.

Table 1. PLL design specification.

PLL design specification	
PLL type	PFD using D-FF
Lock time	0.2 µs
Dynamic (tuning) range	81.5 MHz – 440 MHz
Output frequency	95 MHz
Average power consumption	2.47 mW
Transistor count	NMOS-31 PMOS-32

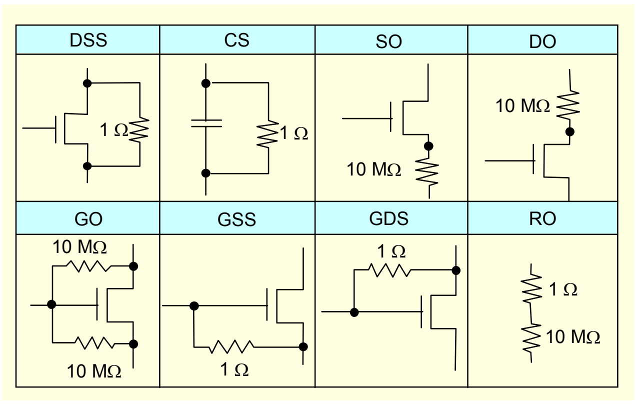

Fig. 7. Fault models.

VII. Fault Models

Figure 7 shows the various faults models that were introduced for each transistor in the PLL as the following:

- DSS: drain source short
- GSS: gate source short
- GDS: gate drain short
- GO: gate open
- DO: drain open
- SO: source open
- RO: resister open
- CS: capacitor short

Low resistance (1 Ω) and high resistance (10 MΩ) were used to model the faults. Shorts were modeled using low resistance, and the open circuits were modeled using high resistance. The faulty and fault free PLLs were designed using 0.18 μm TSMC technology with 2.5 V supply voltage.

VIII. Monte Carlo Analysis

Monte Carlo analysis performs the simulation runs using randomly chosen parameter values. The parameter values for each run are chosen from probability distributions. Monte Carlo analysis sweeps parameter values that are chosen based on statistical variations. Hence the Monte Carlo simulation is used to measure the parametric variations of the faulty and fault free PLL circuits. Monte Carlo analysis was performed for the variations of the threshold voltage in the 0.18 μm TSMC model file using T-spice. Monte Carlo analysis was performed for the threshold voltage of all the NMOS and PMOS transistors with uniform distribution centered at 0.3694291 and 0.3944719 with relative variations of 5%. A total of 20 Monte Carlo analyses of all the 379 faulty and fault free circuits were performed. The measurement results were obtained along with the statistical results from the analyses (namely, minimum, maximum, mean, average deviation, variance, and sigma). Then, these results were used as the data sets for the training and testing of the BPNN.

IX. Back Propagation Neural Network

The back propagation method is a technique used in training multilayer neural networks in a supervised manner. The back propagation method, also known as the error back propagation algorithm, is based on the error-correction learning rule. It consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern is applied to the input nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass, the synaptic weights of the networks are all fixed. During the backward pass, the synaptic weights are all adjusted in accordance with an error-correction rule. The actual response of the network is subtracted from a desired response to produce an error signal. This error signal is then propagated backward through the network. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The weight adjustment is made according to the generalized delta rule to minimize the error. An example of a three layer BPNN with one hidden layer is shown in Fig. 8.

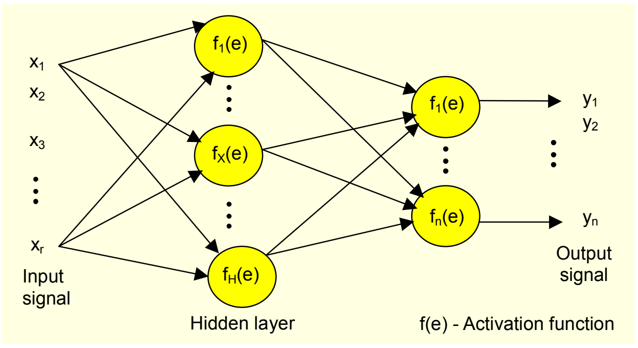

Fig. 8. Back propagation neural network.

Two commonly used neuron activation functions for the neuron in Fig. 8 are sigmoidal and tansig functions. Both functions are continuously differentiable everywhere and typically have the following mathematical forms:

$$\text{Log sigmoidal:}\quad f(e)=\frac{1}{1+\exp(-ae)},\quad a>0, \qquad (1)$$

$$\text{Pure linear:}\quad f(e)=\beta e,\quad \text{for}\quad \beta>0. \qquad (2)$$

X. Types of BPNN Training Algorithms Used for Fault Classification

The back propagation neural networks are trained with nine different training algorithms.

1. Variable Learning Rate BP with Momentum (traingdx)

The learning rate parameter is used to determine how fast the BPNN method converges to the minimum solution. The higher the learning rate, the bigger the step and the faster the convergence. However, if the learning rate is made too high the algorithm will become unstable. On the other hand, if the learning rate is set too low, the algorithm will take a long time to converge. To speed up the convergence time, the variable learning rate gradient descent BP utilizes higher learning rate α when the neural network model is far from the solution and smaller learning rate α when the neural net is near the solution. The new weight vector W_{k+1} is adjusted in the same way as in the gradient descent with momentum, but with a varying α_k. Typically, the new weight vector W_{k+1} is defined as

$$W_{k+1} = W_k - \alpha_{k+1} g_k + \mu W_{k-1}, \quad (3)$$

$$\alpha_{k+1} = \beta \alpha_k, \quad (4)$$

where β is 0.7 if the new error is greater than 1.04 (old error), β is 1.05 if the new error is less than 1.04 (old error), μ is the momentum factor, and g_k is the gradient.

2. Conjugate Gradient BP (CGP)

The basic BP algorithm adjusts the weights in the steepest descent direction. This is the direction in which the performance function decreases most rapidly. Although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In conjugate gradient algorithms a search is performed along conjugate directions, which generally produces faster convergence than the steepest descent directions. In the conjugate gradient algorithms the step size is adjusted at each iteration. A search is made along the conjugate gradient direction to determine the step size which will minimize the performance function along that line. There are four types of conjugate gradient algorithms which can be used for training. All of the conjugate gradient algorithms start out by searching in the steepest descent direction (negative of the gradient) on the first iteration:

$$p_0 = - g_0, \quad (5)$$

where p_0 is the initial search gradient, and g_0 is the initial gradient.

A line search is then performed to determine the optimal distance to move along the current search direction:

$$x_{k+1} = - x_k + \alpha_k p_k \quad (6)$$

where x_k is the current weight vector, x_{k+1} is the next weight vector, α_k is the learning rate, and p_k is the current search direction. Then, the next search direction is determined so that it is conjugate to previous search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction:

$$P_k = - g_k + \beta_k p_{k-1}, \quad (7)$$

where P_k is the current search direction, and P_{k-1} is the previous search direction.

The various versions of conjugate gradient are distinguished by the manner in which the constant β_k is computed.

A. Fletcher-Reeves (traincgf)

For the Fletcher-Reeves (traincgf) update the procedure is

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}. \quad (8)$$

This is the ratio of the norm squared of the current gradient to the norm squared of the previous gradient.

B. Polak and Ribiere (traincgp)

For Polak and Ribiere (traincgp) the constant β_k is computed as

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}}. \quad (9)$$

C. Powell and Beale (traincgb)

For all conjugate gradient algorithms, the search direction will be periodically reset to the negative of the gradient. The standard reset point occurs when the number of iterations is equal to the number of network parameters (weights and biases), but there are other reset methods that can improve the efficiency of training. One such reset method was proposed by Powell and Beale (traincgb). For this technique we restart if there is very little orthogonality left between the current gradient and the previous gradient. This is tested with the following inequality:

$$\left| g_{k-1}^T g_k \right| \geq 0.2 \, \| g_k \|^2. \quad (10)$$

If this condition is satisfied, the search direction is reset to the negative of the gradient.

D. Scaled Conjugate Gradient Algorithm (trainscg)

Each of the conjugate gradient algorithms that has been discussed so far requires a line search at each iteration. This line search is computationally expensive, since it requires the network response to all training inputs to be computed several times for each search. The scaled conjugate gradient algorithm (trainscg), developed by Moller was designed to avoid the time-consuming line search.

3. Quasi-Newton BP (trainbfg)

Newton's method is an alternative to the conjugate gradient methods for fast optimization. Newton's method often converges faster than conjugate gradient methods. The weight update for the Newton's method is

$$W_{k+1} = W_k - A_k^{-1} g_k, \quad (11)$$

where A_k is the Hessian matrix of the performance index at the current values of the weights and biases. When A_k is large, it is complex and time consuming to compute W_{k+1}. Fortunately, there is a type of algorithm based on the works of Broyden, Fletcher, Goldfarb, and Shanno (BFGS) which is based on Newton's method but which does not require intensive calculation as it does not require calculation of second derivatives. It updates an approximate Hessian matrix at each iteration of the algorithm and computes the update as a function of gradient. This new class of method is called the quasi-Newton method. The new weight W_{k+1} is computed as a function of the gradient and the current weight W_k.

4. One-Step Secant BP (trainoss)

Since the BFGS algorithm requires more storage and computation for each iteration than the conjugate gradient algorithms, there is need for a secant approximation with smaller storage and computation requirements. The one-step secant (OSS) method is an attempt to bridge the gap between the conjugate gradient algorithms and the quasi-Newton (secant) algorithms. This algorithm does not store the complete Hessian matrix; it assumes that at each iteration the previous Hessian was the identity matrix. This has the additional advantage that the new search direction can be calculated without computing a matrix inverse.

5. Levenberg-Marquardt BP (trainlm)

Like the quasi-Newton methods, the Levenberg-Marquardt algorithm [9] was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$$H = J^T J, \tag{12}$$

where H is the Hessian matrix, and J is the Jacobian matrix that contains first derivatives of network errors, and the gradient can be computed as

$$g = J^T e, \tag{13}$$

where the Jacobian matrix contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard back propagation technique that is much less complex than computing the Hessian matrix. The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} =- x_k - [J^T J \ \ + \mu I]^{-1} J^T e, \tag{14}$$

where I is the identity matrix.

When the scalar μ is zero, this is just Newton's method using the approximate Hessian matrix. When μ is large, this becomes a gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

6. Resilient Back Propagation (trainrp)

Multilayer networks typically use sigmoid transfer functions in the hidden layers. These functions are often called "squashing" functions, since they compress an infinite input range into a finite output range. Sigmoid functions are characterized by the fact that their slope must approach zero as the input increases. This causes a problem when using the steepest descent to train a multilayer network with sigmoid functions since the gradient can have a very small magnitude. Small changes are caused in the weights and biases, even though the weights and biases are far from their optimal values.

The purpose of the resilient back propagation (rprop) training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update. The magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value is decreased whenever the derivative with respect to that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same. Whenever the weights oscillate the weight change is reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change will be increased.

XI. Data Set

The total transistor count of the PLL circuit is 63. Six fault models are introduced for each transistor. Altogether, 378 unique faults are introduced in the circuit. In 13 transistors, the drain of each transistor is connected to the source of another transistor; hence, the open drain of one transistor is equivalent to the open source of the other. Thus, the faults of these 13 transistors are neglected, and the remaining 300 transistor faults of the other 50 transistors are considered for analysis as shown in Table 2.

Among these 300 faults, there are 50 unique faults, such as drain open, source open, gate open, drain source short, drain gate short, and gate source short. Fifty faults are randomly mixed to form 2 data sets. As a result, there are 6 pairs of data sets, one for each fault. A pair of fault free data sets is also

Table 2. Total transistors and faults introduced.

Type of transistor	Number of transistors	Number of faults introduced
NMOS	31	6 × 31
PMOS	32	6 × 32
Total	63	378
Considered	50	300

Table 3. BPNN parameter variations.

Range of parameter variation	
Learning rate	0.3 to 0.8
Hidden layer neurons	10 to 15
Epochs for training	100 to 1,500

constructed; hence, we have 7 pairs of data sets, including one pair for each fault and one that is fault free. The fourteen data sets are split into 7 test sets and 7 training sets for the BPNN.

XII. Neural Network Parameters

The following algorithms were used to train BPNN: trainlm, trainbfg, trainrp, trainscg, traincgb, taringcgf, traincgp, trainoss and traingdx. The performance of these training algorithms was compared by varying the following parameters: learning rate, number of neurons in hidden layer, and number of epochs. A high learning rate leads to rapid learning, but the weights oscillate. A lower learning rate leads to slower learning and, hence, an optimized learning rate. The stopping criteria were selected using extensive MATLAB simulation. The ranges of variation in the parameters are shown in Table 3.

For learning rates between 0.3 and 0.8, BPNN gave the best fault classification percentage with all the training algorithms. When the number of neurons in the hidden layer was reduced to below 10, the percentage of fault coverage of the neural network tended to be reduced. Graphic representation of fault coverage when hidden layer neurons were fewer than 10 is shown in Fig. 9**.**

XIII. Fault Coverage Using BPNN

The BPNN was trained with all nine training algorithms and each gave better fault classification coverage at a specific learning rate, number of hidden layer neurons, and number of epochs. These values were derived through extensive MATLAB simulations, and the results are reported in Table 4.

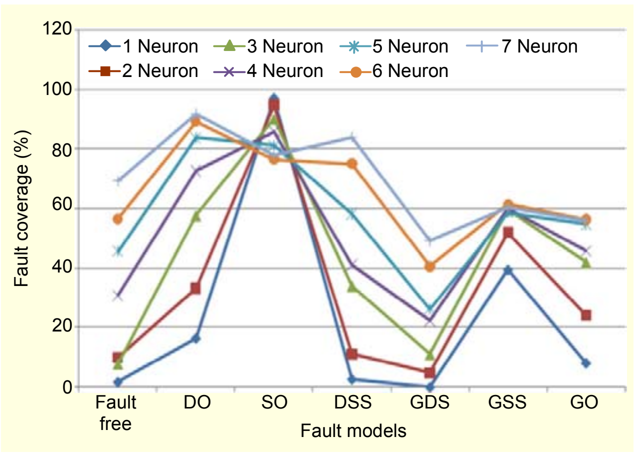

Fig. 9. Consistency of BPNN based on hidden layer neurons.

Table 4. Fault classification of BPNN with various training algorithms.

Training algorithm	Learning rate (α)	Hidden layer neurons (H)	Epochs for convergence	Fault coverage for each fault (%)							Overall fault coverage (%)
				Fault free	DO	SO	DSS	GDS	GSS	GO	
trainlm	0.5	11	500	98	100	100	99	100	98.5	100	99.58
trainrp	0.6	12	800	80	90	100	100	98.58	98	98	97.43
trainbfg	0.7	13	1,400	80	96	85.5	83.18	100	89	100	92.28
trainscg	0.6	15	500	100	98.10	100	95.3	100	100	58	91.89
traincgb	0.6	15	900	68	96	96	72.64	95	95	94	91.44
traincgf	0.3	15	600	100	97	99	100	100	79.2	61.14	89.39
traincgp	0.8	13	700	100	98.5	100	100	84.3	97.40	68.14	91.39
trainoss	0.3	11	1,500	98	100	100	100	100	74.08	78.1	92.03
traingdx	0.5	15	1,500	100	100	100	100	88.30	100	59.44	91.29

Table 5. Performance comparison of BPNN with other techniques.

Technique	Fault coverage (%)
F. Azais et al. [10]	87.7
J.L. Rossello et al. [11]	96.5
C.L. Hsu et al. [12]	97.2
This work	99.58

The results show that trainlm gives 99.58% fault coverage at a 0.5 learning rate, 11 hidden layer neurons, and 500 epochs. This is the best performance in comparison to all other training algorithms.

A comparison of the fault coverage of the BPNN with that of other techniques is presented in Table 5. The proposed BPNN-based fault classification technique has the highest fault coverage results.

XIV. Conclusion

In this paper, we presented a charge-pump PLL which we designed and simulated. Then the output was validated. PLL circuit fault models were introduced, and their Monte Carlo simulated outputs were obtained for training and testing the BPNN. The data localization problem is overcome by the BPNN by randomly picking the training data set. Various training algorithms were used to train the BPNN by varying the learning rate, the number of hidden layer neurons, and the number of epochs. In comparison with other training algorithms, trainlm exhibited the best performance, providing 99.58% fault coverage. The results demonstrate that the BPNN trained with the trainlm algorithm gives best fault classification coverage for CP-PLL circuits.

References

[1] D. Grzechc, J. Ruttkowski, and T. Golonek, “Analog Fault AC Dictionary Creation: The Fuzzy Set Approach,” *Proc. IEEE ISCAS*, May 2006, pp. 5744-5747.

[2] 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, no. 1, Jan. 1993, pp. 102-113.

[3] Z.R. Yang and M. Zwolinski, “Applying a Robust Heteroscedastic Probilisitic Neural Network to Analog Fault Detection and Classification,” *IEEE Trans. Computer Aided Design of Integrated Circuits an Systems*, vol. 19, no. 1, Jan. 2000, pp. 142-151.

[4] V. Stopjakova, P. Malosek, and V. Nagy, “Neural Network-Based Defect Detection in Analog and Mixed IC Using Digital Signal Preprocessing,” *J. Electrical Engineering*, vol. 57, no. 5, 2006, pp. 249-257.

[5] D. Grzechc and J. Ruttkowski, “New Concept to Analog Fault Diagnosis by Creating Two Fuzzy-Neural Dictionaries Test,” *Proc. IEEE MELECON*, May 2004, pp. 115-118.

[6] R.J. Baker, H.W. Li, and D.E. Boyce, *CMOS Circuit Design, Layout and Simulation*, 2nd ed., Prentice Hall of India Pvt. Ltd., 2005.

[7] F.M. Gardner, “Charge-Pump Phase-Locked Loops,” *IEEE Trans. Communications*, vol. COM-28, no. 11, Nov. 1980, pp. 1849-1858.

[8] P. Goteti, G. Devarayanadurg, and M. Soma, “DFT for Embedded Charge-Pump PLL Systems Incorporating IEEE 1149.1,” *Proc. IEEE CICC*, May 1997, pp. 210-213.

[9] S.M.A. Burney, T.A. Jhilani, and C. Ardil, “Levenberg-Marquardt Algorithm for Karachi Stock Exchange Share Rates Forecasting,” *Proc. of World Academy of Science, Engineering, and Technology*, vol. 3, Jan. 2005, pp. 171-176.

[10] F. Azais et al., “A Unified Digital Test Technique for PLLs: Catastrophic Faults Covered,” *Proc. IEEE Int. Mixed Signal Testing Workshop*, 1999, pp. 269-292.

[11] J.L. Rossello and J.Segura, “An Analytical Charge-Based Compact Delay Model for Submicrometer CMOS Inverters,” *IEEE Trans. Circuits and Systems I: Fundamental theory and Applications,* vol. 51, no. 7, July 2004, pp. 1301-1311.

[12] C.L. Hsu, Y. Lai, and S.W. Wang, “Built-in Self Test for Phase-Lock Loops,” *IEEE Trans. Instrumentation and Measurement*, vol. 54, no. 3, June 2005, pp. 996-1002.

Jayabalan Ramesh received the BE degree in electronics and communication engineering in 1995 from Bharathiyar University, Coimbatore, Tamil Nadu, India. He completed his ME degree in communication systems in 1997 from National Institute of Technology, Trichy, Tamil Nadu, India. His research interests include design and testing of analog and mixed signal integrated circuits. He is currently working as a senior lecturer in the ECE department of PSG College of Technology, Coimbatore, India. He is also currently doing his PhD under the guidance of Dr. K. Gunavathi, professor with the ECE Department of PSG College of Technology, Coimbatore India. He is a life member of ISTE, SSI, and IAENG. He has previously published in one international journal and 20 national and international conference publications.

Ponnusamy Thangapandian Vanathi received the BE degree in electronics and communication engineering, the ME degree in computer science and engineering, and the PhD in 1985, 1991, and 2002, respectively, from PSG College of Technology, Coimbatore, Tamil Nadu, India. Her research interests include soft computing, speech signal processing and VLSI Design. She is currently working as an assistant professor in the ECE department of PSG College of Technology, Coimbatore, Tamil Nadu, India. She has around 18 years of teaching and research experience. She is a life member of ISTE. She has published in 10 national and international journals and 50 national and international conference publications.

Kandasamy Gunavathi received the BE degree in electronics and communication engineering, the ME degree in computer science and engineering, and the PhD in 1985, 1989, and 1998, respectively, from PSG College of Technology, Coimbatore, Tamil Nadu, India. Her research interests include low-power VLSI design, design and testing of digital, analog, and mixed signal VLSI circuits. She is currently working as a professor in the ECE department of PSG College of Technology, Coimbatore, Tamil Nadu, India. She has around 20 years of teaching and research experience and is a life member of ISTE. She has published in 20 national and international journals and 60 national and international conference publications.