The Application of Neural Networks to EM-Based Simulation and Optimization of Interconnects in High-Speed VLSI Circuits

Anand Veluswami, *Member, IEEE,* Michel S. Nakhla, *Senior Member, IEEE,* and Qi-Jun Zhang, *Senior Member, IEEE*

(Invited Paper)

***Abstract*—In this paper, a neural network based approach to the electromagnetic (EM) simulation and optimization of high-speed interconnects is discussed. Traditional techniques used to model interconnects in high-speed very large scale integration (VLSI) circuits are based on EM-field simulation, and are thus highly demanding on central processing unit (CPU) resources. This limits their suitability for computer-aided design (CAD) and optimization techniques which are, in general, iterative in nature. Neural networks can be used to map the complex relationship between the physical and electrical parameters of interconnect structures in an efficient manner. The models, once developed, operate with minimal on-line CPU resources and are thus ideally suited for use in iterative CAD and optimization routines.**

***Index Terms*—Modeling, neural networks, optimization, simulation, VLSI interconnects.**

I. Introduction

THE SIMULATION and optimization of the interconnect structures in any high-speed digital system are an essential part of design and optimization in order to ensure proper performance. Several important signal integrity characteristics, such as signal propagation delay, crosstalk, and ground-bounce noise have been identified to be dependent on the interconnect networks and circuits present in the system [1]–[3].

High-speed interconnect analysis is at present a highly central processing unit (CPU) intensive task, characterized by long run-times and large memory requirements. This is due to two main factors. Firstly, the use of an interconnect model in a computer-aided design (CAD) or optimization routine is highly repetitive, as the number of interconnects in any very large scale integration (VLSI) system is extremely large. This is compounded by the fact that most CAD/optimization methods currently used are based on iterative techniques, where a given circuit, or objective function is repeatedly evaluated on-line until an optimal solution is obtained. Typical examples of such techniques, as described in papers such as [4]–[6], and implemented in tools such as [7], are simulated annealing, l_p-based optimization, gradient-based methods, Monte Carlo techniques, statistical/yield analysis, etc. Thus, it is apparent that the on-line time required for the convergence of these techniques is dependent on the efficiency and run-time of models employed.

Secondly, interconnect analysis at high frequencies must be done using distributed-parameter models, such as the transmission-line equation, as lumped-element techniques cease to be accurate. Distributed parameter models are based on the per-unit-length resistance, inductance, capacitance, and conductance (RLCG) matrices of the interconnect structure. These parameters are not constant at relatively high frequencies, but are frequency-dependent, and must be determined from the physical structure of the interconnects before the transmission-line equation models can be applied [8]. This is referred to as *modeling* of interconnects. Once these parameters are determined, the high-speed circuit can be simulated to obtain its signal integrity characteristics such as propagation delay and crosstalk. This is referred to as *simulation* of interconnects, and is done at the level of the circuit treating interconnects as distinct components between the driving sources and terminations which comprise the high-speed system.

Modeling of lossy interconnects is done by electromagnetic (EM) simulation techniques, which involve the numerical simulation of Maxwell's equations or variants thereof. Full-wave three-dimensional (3-D) EM analysis, being approximation free, gives very accurate results, but is highly CPU-intensive, and thus is not feasible for on-line use in large scale CAD and optimization techniques. Techniques used are extensively described in [8]–[12].

Much research has gone into the electrical simulation of interconnects, and several techniques have been developed. These include the resistance–capacitance (*RC*) tree representation of the interconnect, and the two-pole approximation method [13], [14]. But as these are based on approximations and have underlying assumptions, they compromise accuracy in the process of providing speed. This renders them undesirable from an optimization point of view, as the results of optimization are dependent upon the accuracy of the models employed. Recent advances in simulation techniques such as

Manuscript received June 3, 1996; revised November 8, 1996. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada, and in part by Micronet, a Canadian Network Centre of Excellence on Microelectronics Devices, Circuits, and Systems.

A. Veluswami is with Enabling Technologies, Mitel Corporation, Kanata, Canada K2K 1X3.

M. S. Nakhla and Q. J. Zhang are with the Department of Electronics, Carleton University, Ottawa, Ont., Canada K1S 5B6.

Publisher Item Identifier S 0018-9480(97)03094-9.

[15]–[19], have substantially lowered the run-time associated with interconnect analysis, but still cannot provide the simulation speed necessary for exhaustive iterative optimization.

The unavailability of interconnect techniques suitable for on-line iterative CAD/optimization routines has prompted the use of many fast, simple on-line techniques such as polynomial curve-fit techniques and look-up tables, based on data obtained by extensive off-line simulation. Papers such as [20], [21] report the use of empirical models and curve-fit techniques in interconnect analysis. However, curve-fit techniques are generally capable of handling only mild nonlinearity, and a few variables at a time.

Table look-up techniques have also been used to reduce the on-line CPU requirements of interconnect analysis [22]. These techniques are fast, as the on-line time requirement is the trivial time taken for a query on the table. However, they suffer from the following inherent shortcomings:

- size of a table grows exponentially with the addition of each new variable; hence, the memory requirements are large;
- every entry in the table requires a simulation or measurement to be performed;
- tables are cumbersome to maintain and upgrade, and hence, are not very robust. This is because the size of a table depends on the input space, and that new parameters cannot be added easily. To increase the accuracy, the number of points must be increased so there is a larger number of input points defining each input dimension;
- interpolation of points in a look-up table is a highly localized operation, where only data points in the neighborhood of the value required are used, as opposed to the entire surface of the input–output (I/O) relationship.

In this paper, a neural network approach to the EM simulation and optimization of high-speed interconnects and interconnect circuitry is presented. Neural networks have been applied to several design problems in CAD and modeling in the recent past, as reported in [23]–[28]. In [23], neural network applications such as modeling the characteristics of electronic devices, and the statistical analysis and optimization of circuits were treated. Others such as [24], [25] have shown that problems involving static electrical characteristics and those with a relatively small input space can be effectively handled by the neural network technique. The intent of this paper is to demonstrate that neural networks are also highly suitable for EM-based simulation and optimization, especially given the lack of efficient on-line techniques in this area and the immediacy of the problem. The low run-time and memory requirements and relative simplicity in comparison to traditional modeling and simulation approaches suggest that neural networks are a feasible technique for efficient high-speed interconnect modeling.

In the following section, the high-speed interconnect problem is formulated mathematically, first in general terms, based on existing simulation approaches, then in terms of the proposed neural network based approach. Section III describes in detail how the neural network model is implemented, trained, and validated. The training process is treated mathematically and the main issues and parameters associated with neural network techniques are discussed. In Section IV, several examples which cover various issues in CAD and optimization of high-speed interconnects demonstrate how the neural network model developed in Sections II and III is implemented and used. Section V offers an evaluation of the performance of the neural network technique, and a discussion of the results obtained. A brief summary and conclusion are given in Section VI.

II. Problem Formulation: The High-Speed Interconnect Model

The objective is to create a model which can be used on-line in iterative CAD/optimization routines, capable of mapping the relationship between the set of parameters defining the physical configuration of a network or group of interconnects in a VLSI system and its operational characteristics, and the set of parameters which can be used to analyze the signal integrity of the system.

Mathematically,

$$\mathbf{Y} = \mathcal{F}(\mathbf{X}) \tag{1}$$

where $\mathbf{Y}$ is an n-dimensional output vector representing the parameters to be modeled or simulated, such as the RLCG matrices of the network, the signal propagation delays at the terminations, crosstalk, level of ground-bounce noise, etc., and $\mathbf{X}$ is the m-dimensional input vector containing all the variables and parameters necessary to obtain $\mathbf{Y}$. Typical parameters in the input set are the physical dimensions of the interconnects and their dielectric substrate characteristics, topology of the interconnect network, input signal characteristics such as voltage level, frequency of operation (or rise time in the case of a digital signal), termination impedance of the interconnects, etc., as shown in Fig. 1. The input vector $\mathbf{X}$ in traditional techniques is often not an explicit parameterization of all the input variables. This is especially true for variables such as network topology, which is defined implicitly in the netlist of a circuit.

The function $\mathcal{F}$ is that which relates $\mathbf{Y}$ to $\mathbf{X}$ in the model used on-line during CAD or optimization. Ideally, $\mathcal{F}$ should be simple, easy to evaluate, and have low memory requirements. It is in this function that the neural network approach differs from traditional approaches.

A. Traditional Techniques

In traditional modeling and simulation techniques as described in Section I, $\mathcal{F}$ is evaluated using an electrical analysis tool, such as asymptotic waveform evaluation (AWE), numerical inversion of Laplace transforms (NILT), etc., or EM simulation. The relationship is often not explicit, and can involve netlisting, numerical simulation, and the extraction or separate calculation of the output. For example, if propagation delay or signal ringing is a parameter in $\mathbf{Y}$, it is calculated after transient analysis of the circuit is performed. $\mathcal{F}$ can also be a polynomial relationship in curve-fit techniques, or query, or search routine on a look-up table.

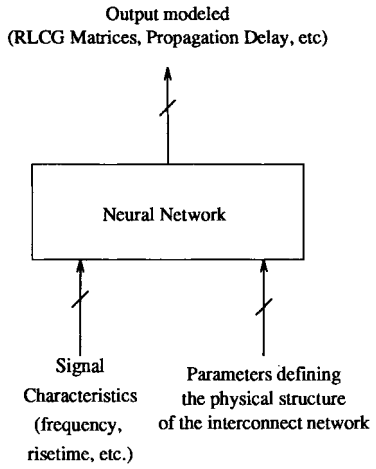

Fig. 1. General structure of the model to be developed.

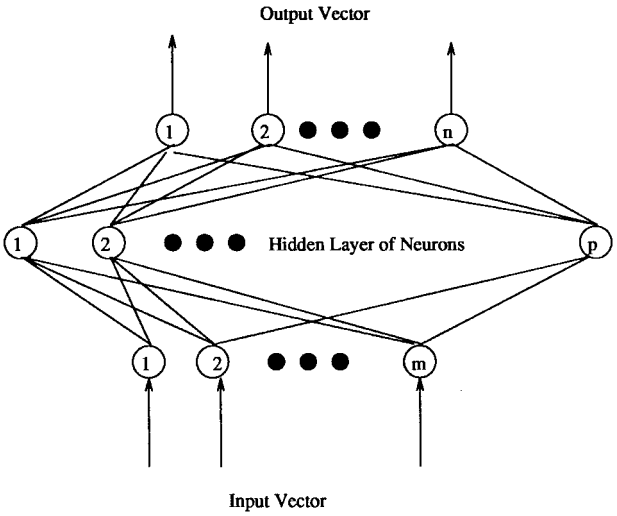

Fig. 2. The neural network model. All inter-neuronal connections are made with weights and all neurons have biases.

B. The Neural Network Approach

In the neural network approach, the function $\mathcal{F}$ is mapped using a neural network.

A feed forward neural network can be described as a mathematical tool which is capable of nonlinear mapping in high dimension [29]. The input space, $\mathbf{X}$ of dimension m is mapped to the n-dimensional output space represented as a layer of n neurons, through a hidden layer. This hidden layer has a fixed number of neurons, p, which can vary from problem to problem as will be discussed in the following section. The output of any given neuron is the weighted linear combination of the outputs of all the neurons in the previous layer reflected off a nonlinear transfer function, the most commonly employed being the sigmoid. The neural network is pictorially represented in Fig. 2.

Mathematically, the neural network can be described as the mapping of the set of input vectors $\mathbf{X}$, whose kth sample is

$$\mathbf{x}_k = (x_{k1}, \quad x_{k2}, \quad \cdots, \quad x_{km}) \tag{2}$$

to the corresponding output vector

$$\mathbf{y}_k = (y_{k1}, \quad y_{k2}, \quad \cdots, \quad y_{kn}) \tag{3}$$

through a system of weighting factors and biases, which are defined as w_{ih}, b_h, for $i = 1, 2, \cdots, m$ and $h = 1, 2, \cdots, p$, and v_{hj}, c_j, for $h = 1, 2, \cdots, p$ and $j = 1, 2, \cdots, n$, such that the n outputs are

$$y_{kj} = f(\zeta_{kj}) = \frac{1}{1 + e^{-\zeta_{kj}}} \tag{4}$$

where

$$\zeta_{kj} = \sum_{h=1}^{p} z_{kh} v_{hj} + c_j. \tag{5}$$

Here, $f(\zeta)$ is the sigmoidal transfer function, and z_{kh} is the output of the hth neuron in the hidden layer, calculated as

$$z_{kh} = f(\gamma_{kh}) = \frac{1}{1 + e^{-\gamma_{kh}}} \tag{6}$$

where

$$\gamma_{kh} = \sum_{i=1}^{m} x_{ki} w_{ih} + b_h. \tag{7}$$

It is seen that $\mathbf{X}$ is now related to $\mathbf{Y}$ by a set of sample data. If the set of samples $\mathbf{x}_k$, $k = 1, 2, \cdots, Q$ is chosen such that it is representative of the entire I/O space, then the objective mapping function $\mathcal{F}$ is in effect learned by the neural network. Since the relationship expressed in (4)–(7) has only used two basic arithmetic operations, i.e., the sum of products and exponentiation, the run-time required to calculate $\mathcal{F}(\mathbf{X})$ is trivially small.

III. Implementation of the Neural Network Model

A. Configuration of the Neural Network Model

The three components of the neural network model are as follows.

1) The Input Layer: The input layer of the neural network consists of m nodes, representing the elements of the $\mathbf{X}$ vector. This must contain all the necessary information and variables required to uniquely map $\mathcal{F}$, in an explicitly parameterized form. In particular, variables such as the layout topology of an interconnection network, which are implicit in netlist approaches, must be quantified numerically.

The authors have chosen to quantify network topology using graph theory, by representing the interconnection network as a minimum spanning tree, rooted at the source, or any other pin of interest. The set of variables which represent the nodes at which a given interconnect has commenced (the source vertices of the edges of the rooted spanning tree describing the network) uniquely represents the network configuration [27].

2) The Output Layer: The output layer consists of n neurons, each representing one of the elements in the $\mathbf{Y}$ vector. The output of a neuron varies between 0 and 1, which are the asymptotic limits of the sigmoid transfer function, as given in (5). Hence, the entire output space must be scaled to vary between 0 and 1.

3) The Hidden Layer: The number of neurons in the hidden layer is taken to be p, the choice of the value of p being dictated by the complexity of the problem. A highly complex I/O relationship would require a higher number of neurons in the hidden layer, as each additional neuron provides an additional degree of freedom during mapping. However, a neural network with a large value of p would require a larger training time, and would unnecessarily increase the size $\mathcal{S}$ of the model, measured as the total number of weights and biases in the neural network

$$\mathcal{S} = p(m+1) + n(p+1). \tag{8}$$

So as small a p as would allow for correct mapping of the I/O relationship is used.

a) The size of the hidden layer: Deciding the size of the hidden layer is a critical part of the designing of a neural network model. Unfortunately, there are no established methods to decide the appropriate number of hidden neurons required for a given problem. In general, for the models developed in this paper, the number of hidden neurons was decided based on the following:

- size of the input space, which was more critical than the size of the output space (especially when the outputs vary with respect to the inputs in a similar fashion). As will be seen in the examples, Examples A–C have the same number of hidden neurons, even though the number of outputs increases;
- complexity of the problem. The problem mapped in Example D is more complex than the earlier ones, due to the fact that it has both continuous and discrete inputs.

B. Training

The training of the neural network is the process during which the neural network learns the relationship $\mathcal{F}$ between the input and output samples presented to it. This relationship is learned over several *training epochs*, in which a large set of I/O data is repeatedly presented to the neural network. The weights and biases in (4)–(7) are adjusted such that the error between the outputs as predicted by the neural network and the outputs of the training set is minimized. The algorithm employed is based on the classical multilayered backpropagation algorithm, and is described as follows. The algorithm has been described in [23], and can also be found in several reference books such as [29]. It is briefly outlined here for the sake of completeness.

For a given set of input data, say $\mathbf{a}_k$, $k = 1, 2, \cdots, Q$ whose corresponding output set is $\mathbf{d}_k$, if the neural network predicts the output to be $\mathbf{y}_k$, the batch-mode backpropagation error E_{av} is defined as

$$E_{av} = \sum_{k=1}^{Q} E_k = \frac{1}{Q} \sum_{k=1}^{Q} \left[\frac{1}{2} \sum_{j=1}^{n} (y_{kj} - d_{kj})^2 \right] \tag{9}$$

where E_k represents the individual mean-squared error of the kth sample. E_{av} is the error to be minimized during training. After each training epoch, during which the set of Q data points is presented to the network, this error is determined, and the weights and biases are updated in the general direction of error minimization. The update equations for the tth epoch, with momentum α and learning rate η are

$$v_{hj}^{t+1} = v_{hj}^{t} + \eta(E_{av}^{t}).\frac{\partial E_{av}^{t}}{\partial v_{hj}} + \alpha.(v_{hj}^{t} - v_{hj}^{t-1}) \tag{10}$$

$$w_{ih}^{t+1} = w_{ih}^{t} + \eta(E_{av}^{t}).\frac{\partial E_{av}^{t}}{\partial w_{ih}} + \alpha.(w_{ih}^{t} - w_{ih}^{t-1}) \tag{11}$$

$$b_{h}^{t+1} = b_{h}^{t} + \eta(E_{av}^{t}).\frac{\partial E_{av}^{t}}{\partial b_{h}} + \alpha.(b_{h}^{t} - b_{h}^{t-1}) \tag{12}$$

$$c_{j}^{t+1} = c_{j}^{t} + \eta(E_{av}^{t}).\frac{\partial E_{av}^{t}}{\partial c_{j}} + \alpha.(c_{j}^{t} - c_{j}^{t-1}). \tag{13}$$

The error sensitivities in the above equations are calculated using the following equations:

$$\begin{aligned}\frac{\partial E_{av}}{\partial v_{hj}} &= \frac{\partial}{\partial v_{hj}} \left[\frac{1}{2Q} \sum_{k=1}^{Q} \sum_{j=1}^{n} (y_{kj} - d_{kj})^2 \right] \\ &= \frac{1}{Q} \sum_{k=1}^{Q} (y_{jk} - d_{jk}).y_{jk}.(1 - y_{jk}).z_{hk} \\ &= \frac{1}{Q} \sum_{k=1}^{Q} \delta_{jk}^{(o)}.z_{hk}\end{aligned} \tag{14}$$

where the term $\delta_{jk}^{(o)}$, given by

$$\delta_{jk}^{(o)} = (y_{jk} - d_{jk}).y_{jk}.(1 - y_{jk}) \tag{15}$$

represents the local gradients at the jth neuron in the output layer for the kth sample

$$\begin{aligned}\frac{\partial E_{av}}{\partial w_{ih}} &= \frac{\partial}{\partial w_{ih}} \left[\frac{1}{2Q} \sum_{k=1}^{Q} \sum_{j=1}^{n} (y_{jk} - d_{jk})^2 \right] \\ &= \frac{1}{Q} \sum_{k=1}^{Q} z_{hk}.(1 - z_{hk}) \sum_{j=1}^{n} (y_{jk} - d_{jk}).\delta_{jk}^{(o)}.a_{ik}\end{aligned} \tag{16}$$

$$\begin{aligned}\frac{\partial E_{av}}{\partial c_{j}} &= \frac{\partial}{\partial c_{j}} \left[\frac{1}{2Q} \sum_{k=1}^{Q} \sum_{j=1}^{n} (y_{jk} - d_{jk})^2 \right] \\ &= \frac{1}{Q} \sum_{k=1}^{Q} (y_{jk} - d_{jk}).y_{jk}.(1 - y_{jk})\end{aligned} \tag{17}$$

$$\begin{aligned}\frac{\partial E_{av}}{\partial b_{h}} &= \frac{\partial}{\partial b_{h}} \left[\frac{1}{2Q} \sum_{k=1}^{Q} \sum_{j=1}^{n} (y_{jk} - d_{jk})^2 \right] \\ &= \frac{1}{Q} \sum_{k=1}^{Q} z_{hk}.(1 - z_{hk}) \sum_{j=1}^{n} (y_{jk} - d_{jk}).\delta_{jk}^{(o)}.\end{aligned} \tag{18}$$

The parameter η, known as the learning rate, is an important training parameter representing the step size of the error convergence process. A small value of η affords stability but increases training time, while a large value of η decreases the stability of the training process. η is shown in the update equations as being a function of the backpropagation error E^t, meaning it is normally adaptive in nature, assuming a small value when far away from the solution, during which the value of E^t is large, and a relatively larger value when closer to the

solution, i.e., when the backpropagation error is small. The value of η is normally taken to be between 0.1–0.5.

The term α in the update equations is called the momentum, and adds an inertial element to the training process, ensuring that the update occurring is a fraction of the previous change. This helps avoiding local minima on the error surface. α normally assumes a positive value less than 1.

Using the above equations, the backpropagation training algorithm can be summarized in the following steps.

Step 1) Initialization: Select random initial values for the weights and biases w_{ih}, b_h, v_{hj}, and c_j lying in the range of -0.5 and 0.5.

Step 2) Presentation: Present the training set $\mathbf{x}_k, k = 1, 2, \cdots, Q$ to the neural network.

Step 3) Forward Pass: Compute the corresponding neural network output vector $\mathbf{y}_k, k = 1, 2, \cdots, Q$ using (4)–(7).

Step 4) Update: Compute the batch backpropagation error from (9), the error sensitivities using (14)–(18), and update the weights using (10)–(13).

Step 5) Termination Condition Check: If E^t is less than a specified tolerance value end training, else update learning rate (if necessary) and return to Step 2.

A more detailed description of the training algorithm can be found from Haykin's book [29]. For the applications presented in this paper, the Matlab implementation of the backpropagation algorithm [30] was employed.

C. Data Generation

In order to train and validate the neural network, two sets of data are required. The first set, namely the training set, is a set of data points representing randomly chosen inputs to the model and their corresponding outputs. The number of data points needed in the training set is dependent upon the following:

- number of neurons in the hidden layer p;
- dimensionality of the problem, particularly dimensionality of the input space;
- complexity of the relationship between input and output.

Though a larger training set would give a better representation of the I/O space to be modeled, too large a training set would result in a needless increase in the time invested in data generation and training. Hence, as small a data set as would fairly represent the entire range of operation of this model is used.

The training set data are obtained through repeated off-line simulation using an accurate simulation technique; either an electrical CAD tool or an EM-field solver, such as those previously described, depending on the nature of the outputs. The simulator is repeatedly called, each time with a netlist incorporating input variables randomly chosen from the input space as shown in Fig. 3. Two files of data are thus obtained, the input vector and the corresponding output vector, which constitute the training set. This is used in training the model. The training set can also be obtained as a collection of data from actual measurements, or from a look-up table if available.

The second set of data, called the test set, is obtained in a manner identical to that described above and is used to test

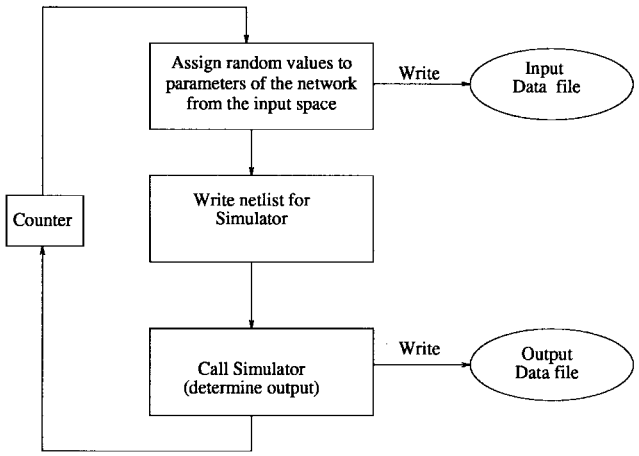

Fig. 3. Algorithm used for generation of training and test data.

the accuracy of the model during and after training. The test set should be large enough to be representative of the entire input space, and its contents should be different from those of the training set. The test set is used solely for the purpose of validation of the model.

1) The Size of Training and Test Data Sets: The sizes of the training and test data sets depend on the size of the neural network, especially the input and hidden layers. These reflect the complexity of the problem. While training proceeds, if it is found that there is insufficient error convergence, or that the test error is significantly higher than the training error, it is indicative that the training set is not large enough, and should be made larger. It is seen that in the models developed in this paper, the size of the output vector is less important than that of the input in determining the size of the training and test data sets. This is seen in Examples A–C. This is because the output characteristics as functions of the input are generally of the same shape, though they differ in quantitative values. So the trend of variation of the output with respect to the input learnt with a smaller number of outputs (Example A) is the same as with a larger number outputs (Example C), so the size of the data sets remained the same.

D. Model Validation

The model is validated at the end of training, as well as periodically during the training process with both the training set and the test set. The rms error for the kth sample of the training or test set is defined as

$$\varepsilon_k = \left[\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_{ki} - d_{ki}}{\overline{d}_i}\right)^2\right]^{1/2} \tag{19}$$

where n is the number of outputs of the neural network, y_{ki} is the output as given by the neural network for the kth sample, d_{ki} is the corresponding actual output of the simulator, and $\overline{d}_i$ is the arithmetic mean of the absolute values of all ith output points over the entire space

$$\overline{d}_i = \frac{1}{Q}\sum_{k=1}^{Q}|d_{ki}|. \tag{20}$$

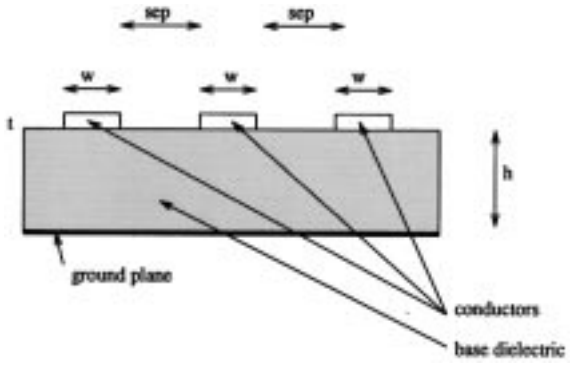

Fig. 4. Cross section of three parallel interconnects (Example A).

The average (test/training) error of validation is given as the arithmetic mean of the individual errors over the (test/training) set, i.e.,

$$\text{Average Error} = \frac{1}{Q}\sum_{k=1}^{Q}\varepsilon_k \tag{21}$$

where Q is the total number of samples in the set. It should be noted that the validation error is different from the backpropagation error E given in (9), which is only used in the backpropagation training algorithm.

The average test error is a good indication of how well the neural network has learned the relationship $\mathbf{Y} = \mathcal{F}(\mathbf{X})$. The neural network is validated periodically during training with the test set to ensure that the network is not *overlearning*, which is when the mapping is being concentrated too much on the specific data points of the training set rather than the general function. When overlearning starts, the average test error begins to increase though the average training error and backpropagation error continue to decrease.

Once developed, the model can be used to accurately calculate the desired output for any given set of inputs.

IV. Applications and Numerical Examples

A. Three Parallel Coupled Interconnects

In this first example, a configuration of three parallel interconnects, the cross section of which is shown in Fig. 4, is modeled for its frequency-dependent $\mathbf{L}$ and $\mathbf{C}$ parameters. During design optimization, each individual configuration would vary in terms of the thickness and width of the interconnects, the height of the dielectric, the separation between the center conductors, and the frequency of operation. Hence,

$$\mathbf{X} = (W, t, s, h, f). \tag{22}$$

The ranges of the individual input variables are given in Table I. The outputs of the model are the elements of the $\mathbf{L}$ and $\mathbf{C}$ matrices

$$\mathbf{L} = \begin{pmatrix} L_{11} & L_{12} & L_{13} \\ L_{21} & L_{22} & L_{23} \\ L_{31} & L_{32} & L_{33} \end{pmatrix}$$

TABLE I
Neural Network Input Parameters and Their Range (Example A)

Parameter	Symbol	Minimum	Maximum
Conductor Width	W	5mil	11mil
Conductor Thickness	t	0.7mil	2.8mil
Separation	s	1mil	16mil
Dielectric height	h	5mil	10mil
Frequency	f	1MHz	8GHz

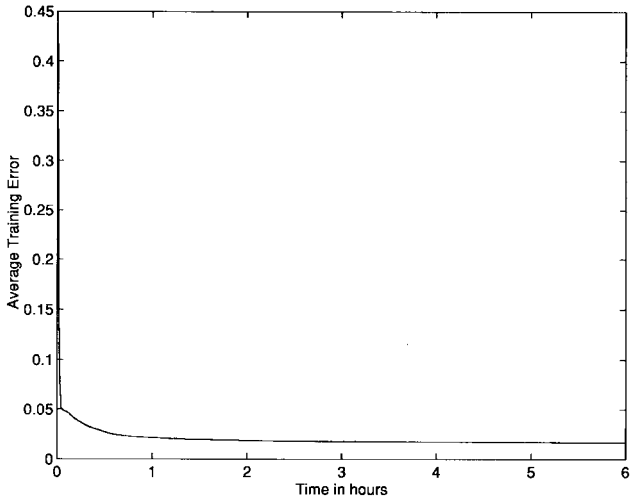

Fig. 5. Average training error as training proceeded for Example A.

and

$$\mathbf{C} = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix}.$$

The total number of $\mathbf{L}$ and $\mathbf{C}$ parameters to be simulated is 18. Since the interconnects are identical in physical dimension and have symmetric cross sections, one needs to consider only L_{11}, L_{12}, L_{13}, C_{11}, C_{12}, and C_{13}, from which the entire $\mathbf{L}$ and $\mathbf{C}$ matrices can be constructed. Hence,

$$\mathbf{Y} = (L_{11}, L_{12}, L_{13}, C_{11}, C_{12}, C_{13}). \tag{23}$$

The other per-unit-length parameters of the structure, $\mathbf{R}$ and $\mathbf{G}$ would be modeled in a manner identical to that of $\mathbf{L}$ and $\mathbf{C}$.

The neural network model is implemented with features as shown in Table II, which shows the number of inputs, outputs, and hidden neurons. The technique used to generate data was by off-line simulation using Northern Telecom's SALI, an EM-field solver [31]. However, any EM technique, such as those described in Section I can be used. Training proceeded on a Sun SPARCstation 10, the test error continually decreasing until it leveled off at about 0.017. The average training error as training proceeded is plotted against time in Fig. 5. The test error, sampled periodically during training, followed the same curve as the training error; however, it was marginally higher. The average test error was 0.0174 at the end of training.

The neural network model was found to estimate the $\mathbf{L}$ and $\mathbf{C}$ parameters with an accuracy comparable to the field solver

TABLE II
FEATURES OF THE NEURAL NETWORK MODELS DEVELOPED

Feature	Example A	Example B	Example C	Example D
Number of inputs (m)	5	6	5	18
Number of outputs to overall model	18	8	128	4
Number of outputs nodes in neural network (n)	6	8	30	4
Number of neurons in hidden layer (p)	10	10	10	40
Size of model (in floating point numbers)	126	158	390	880
Size of training set	500	500	500	4500
Size of test set	500	500	500	4500
Data Generation Technique	SALI	SALI	SALI	NILT
Training time (in hours)	6	7	7	48
Average training error	0.0162	0.0233	0.0348	0.0356
Average test error	0.0174	0.0253	0.0351	0.04

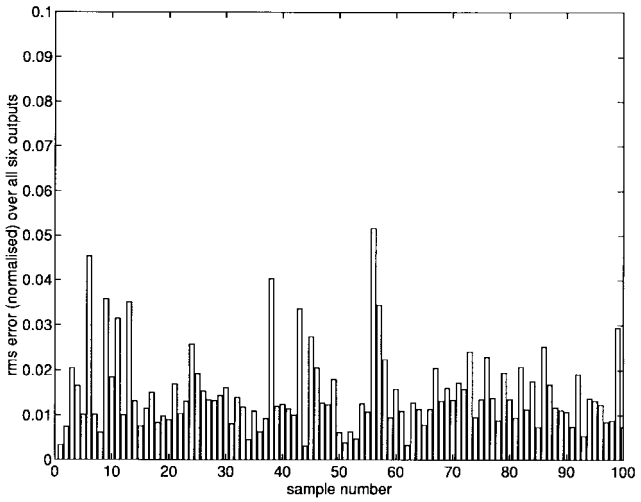

Fig. 6. Test error for 100 random inputs from the test set (Example A).

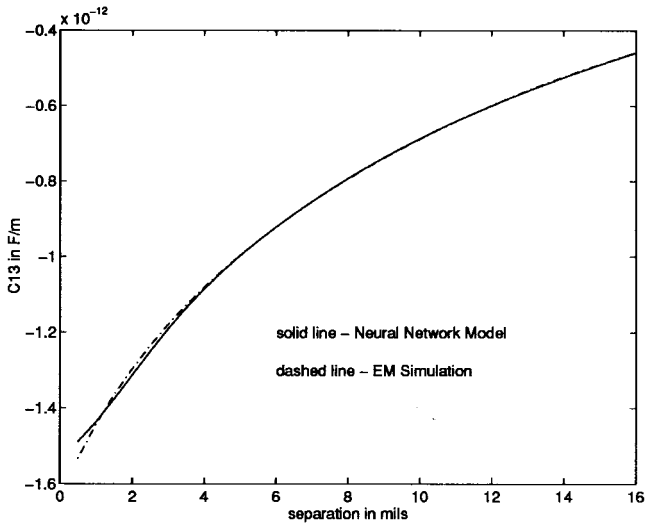

Fig. 7. C_{13} as a function of separation, keeping other parameters constant in Example A.

used. The average test error in the results when these parameters were modeled for 100 random circuit configurations from the test set are plotted in Fig. 6. Fig. 7 shows one of the outputs, C_{13}, plotted against a varying width of separation as estimated by the neural network model and the EM-field solver.

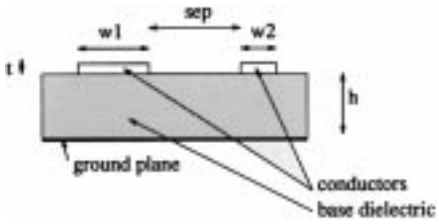

Fig. 8. Two asymmetric microstrip lines (Example B).

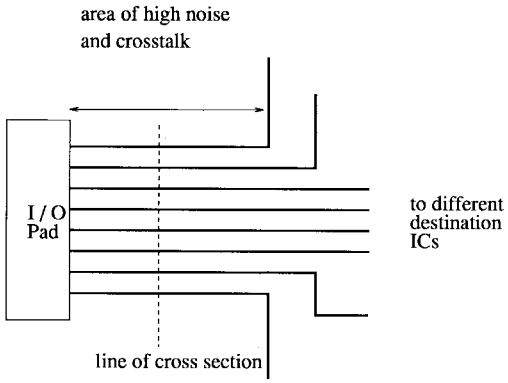

Fig. 9. Eight-bit bus on a printed circuit board (PCB), showing region of high crosstalk and signal noise.

B. Two Asymmetric Interconnects

The $\mathbf{L}$ and $\mathbf{C}$ parameters of an asymmetric configuration of two parallel interconnects, as shown in Fig. 8 are treated in this example. The input variables are identical to those in the previous example except that the width of the two conductors must be given separately

$$\mathbf{X} = (W_1, W_2, t, s, h, f). \tag{24}$$

Their ranges are as given in Table I. As in the previous example, EM simulation is used to obtain the L and C

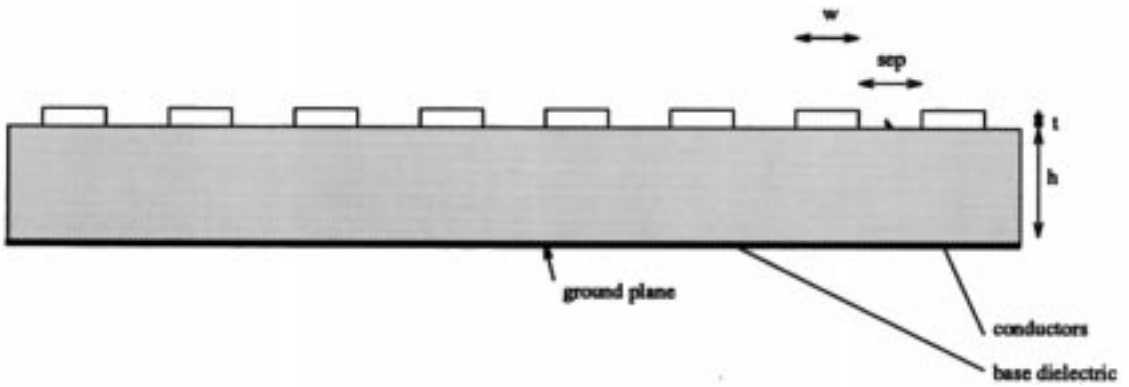

Fig. 10. Cross section of eight-bit digital bus.

parameters during off-line data generation. There are eight outputs, namely the elements of the $\mathbf{L}$ and $\mathbf{C}$ parameter matrices

$$\mathbf{Y} = (L_{11}, L_{12}, L_{21}, L_{22}, C_{11}, C_{12}, C_{21}, C_{22}). \quad (25)$$

Table II shows the features of the neural network developed. The accuracy of the neural network model was in the range of the EM simulator.

C. An Eight-Bit Digital Bus Configuration

In this example, an eight-bit bus configuration as shown in Fig. 9, the cross-sectional view of which is shown in Fig. 10, is modeled. In the region between the I/O pads and the IC pin's to which the inputs are connected, the eight interconnects run parallel to each other. This is a region of high crosstalk and noise due to the eight metallic conductors, and is thus especially critical in layout design.

A digital bus is inherently symmetric in structure, and consists of conductors of identical physical cross section. The inputs of the model and their ranges are exactly the same as in the first example, as given in Table I.

The outputs modeled in this case are the elements of the $\mathbf{L}$ and $\mathbf{C}$ parameter matrices, each of which contains 64 elements. However, since the interconnects are identical in physical dimension, width of separation, and have symmetric cross sections, the 8×8 matrices are symmetric about both diagonals. Thus the entire $\mathbf{L}$ matrix can be constructed from (L_{11}), and 16 other values. Similarly, the $\mathbf{C}$ matrix can be built from (C_{11}), and 16 other elements in the matrix. Further, in the case of the inductance matrix, the values of the elements farthest apart, namely L_{18}, L_{17}, L_{28}, and L_{27} are extremely small and, hence, are set to a nominally low average value. The $\mathbf{L}$ matrix is represented as

$$\mathbf{L} = \begin{pmatrix} L_{11} & L_{12} & L_{13} & L_{14} & L_{15} & L_{16} & \overline{L}_{17} & \overline{L}_{18} \\ L_{12} & L_{11} & L_{23} & L_{24} & L_{25} & \overline{L}_{26} & \overline{L}_{27} & \overline{L}_{28} \\ L_{13} & L_{23} & L_{11} & L_{34} & L_{35} & L_{36} & \overline{L}_{37} & L_{38} \\ L_{14} & L_{24} & L_{34} & L_{11} & L_{45} & L_{35} & L_{25} & L_{15} \\ L_{15} & L_{25} & L_{35} & L_{45} & L_{11} & L_{34} & L_{24} & L_{14} \\ L_{16} & \overline{L}_{62} & L_{36} & L_{34} & L_{35} & L_{11} & L_{23} & L_{13} \\ \overline{L}_{71} & \overline{L}_{72} & \overline{L}_{73} & L_{24} & L_{25} & L_{23} & L_{11} & L_{12} \\ \overline{L}_{81} & \overline{L}_{82} & L_{16} & L_{14} & L_{15} & L_{13} & L_{12} & L_{11} \end{pmatrix} \quad (26)$$

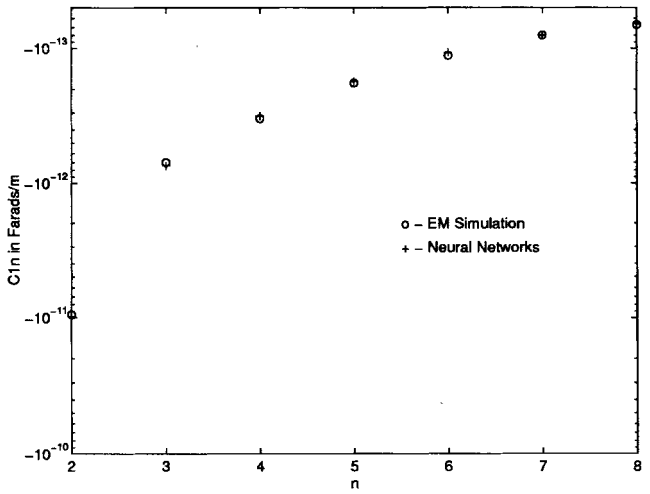

Fig. 11. The values of capacitances between conductors 1 and n (Example C).

where the elements with bars are fixed at nominally average values and not separately modeled. The $\mathbf{C}$ matrix is similar to the $\mathbf{L}$ matrix, except that all parameters are modeled. Hence, the number of outputs for the neural network are 17 C elements and 13 L elements, resulting in a total of 30.

The main features of the neural network developed are tabulated in Table II. It was capable of accurately predicting the L and C parameters. Fig. 11 shows the values of the capacitances in a given configuration as one moves away from the first conductor on the left, i.e., the values on the first row of the $\mathbf{C}$ matrix. Fig. 12 shows the error on 100 random input points of test data.

D. A Network of Four Interconnects on a Printed Circuit Board (PCB)

Here, a model to simulate the propagation delays of a signal propagating in a network of interconnects on a PCB, a typical section of which is shown in Fig. 13, is developed. One IC pin acts as a source, from which a digital signal is transmitted to several other receiver (driven) IC pin's. The number of pin's connected to a given driver is related to the fan-out of a typical pin in a digital IC. This model treats drivers connected to four pin's. The electrical equivalent circuit of the interconnects of Fig. 13 is shown in Fig. 14. A typical

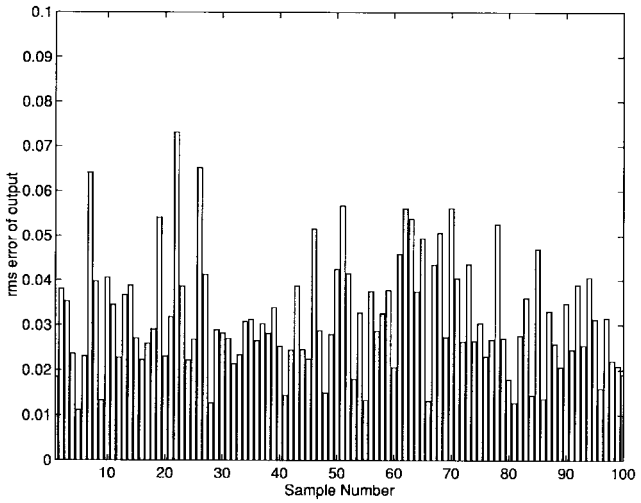

Fig. 12. Test error for 100 random inputs from the test set for Example C.

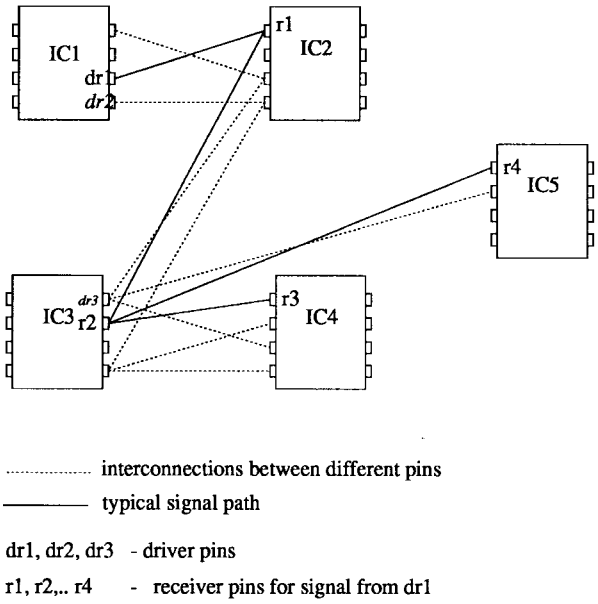

Fig. 13. A typical interconnect network with IC pin's connected by several interconnects.

PCB would consist of several hundreds of such configurations. During layout optimization of such a PCB, each individual interconnect network would vary in terms of its driver (source) characteristics, its receiver pin load characteristics, the lengths of the interconnects, and the network topology. The input variables to this problem along with their ranges of variation are given in Table III:

$$\mathbf{X} = (l_i, R_i, C_i, R_s, r_t, V_p, e_j) \qquad \text{for } i = 1, 2, 3, 4 \text{ and } j = 1, 2, 3. \tag{27}$$

The outputs of the model are the propagation delays at the four terminations, propagation delay being measured as the time taken for the signal to reach 80% of its steady state value

$$\mathbf{Y} = (\tau_i) \qquad \text{for } i = 1, 2, 3, 4. \tag{28}$$

The number of inputs, outputs, and the number of hidden neurons chosen for the neural network model are shown in Table II. Due to the large dimensionality of the problem, as well as the nature of the input variables, the topology variables in particular, a relatively high number of training and test data points is used. Off-line data generation is done using NILT [32], which is a circuit simulation technique employed in several industrial environments. Training proceeded on a Sun SPARCstation 10, the test error continually decreasing until it leveled off at about 0.04.

This example forms the basis of a neural network interconnect simulator which can be used in propagation delay analysis. Based on the same principle, models for different sizes of networks (two to six) were built in [28], and used to construct an interconnect network simulator. Substantial speed-ups were seen to be obtained over existing techniques, especially in an optimization environment when a large number of circuits must be simulated. The speed-ups achieved and their significance with regards to circuit simulation and optimization are discussed in the following section.

V. Discussion of Results

A. Run-Time Comparison

In this section, the on-line run-time requirements of the neural network models are compared with existing simulation techniques.

The run-time requirements are compared with other techniques using the time required to simulate a specific number of circuit configurations. During each iteration of the optimization routine, the model is called once for every occurrence of the structure in the system. A conservative estimate of the number of times an interconnect model is to be called during optimization, in an IC placement-oriented interconnect problem such as described in [6], to obtain a reasonably, if not globally, optimal solution in an optimization problem is taken as 20 000.

In Table IV, the run-time required for estimating the L and C parameters of 20 000 different interconnect structures of the type modeled in Examples A–C, by EM simulation and by the neural network technique are reported. A substantial speed-up is seen.

Table V shows a comparison of run-time for electrical simulation of 20 000 PCB structures as modeled in Example D. The neural network technique is compared to two other currently used simulation techniques, AWE [15] and NILT. The speed-up ratio obtainable by using neural networks over the existing technique is shown in brackets.

B. Performance Evaluation

From the above examples, it is apparent that neural networks overcome many of the limitations of traditional simulation techniques. The most significant advantage is the substantial on-line speed-up offered. This permits designers to be much more liberal in defining critical paths to be tested and optimized, and in the number of iterations that are to be performed in the optimization routine. Using neural networks as a fast, on-line simulation tool can allow exhaustive optimization routines to be performed without too much concern over the

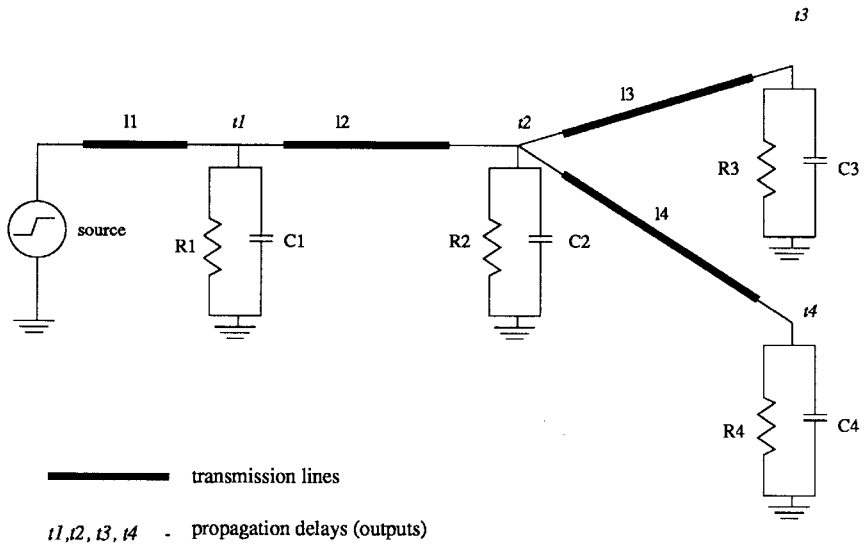

Fig. 14. Electrical equivalent of interconnect configuration, showing the outputs of the model (Example D).

TABLE III
RANGE OF INPUT PARAMETERS TO THE NEURAL NETWORK MODEL

Parameter	Number	Unit	Minimum	Maximum
Interconnect length (l)	4	cm	1	15
Termination Resistance (R_i)	4	Ohms	100	100,000
Termination Capacitance (C_i)	4	nF	3.3	5
Source Resistance (R_s)	1	Ohms	13.3	45
Input Rise Time (r_t)	1	ns	1.6	10
Peak Voltage (V_p)	1	V	.8	5
Source Edges (defining topology)	3	-	2	6

TABLE IV
COMPARISON OF RUN-TIME REQUIREMENTS FOR NEURAL NETWORK MODEL AND EM SIMULATION (EXAMPLE D)

Method	Run-time for 20,000 configurations
EM Simulation	20-80 hours
Neural Network Model	40-130 seconds

TABLE V
COMPARISON OF RUN-TIME FOR DELAY SIMULATION (EXAMPLE D)

Method	Run-time (Speed-up ratio)
NILT	34.43 hours (310)
AWE	9.56 hours (86)
Neural Network Model	6.67 minutes (1)

time required for convergence, which is a major concern with existing techniques.

A neural network can be developed for any defined cross section which might be encountered during layout optimization, irrespective of size or the number of conductors. Memory requirements are not high, and do not grow exponentially with the number of inputs or outputs added as a look-up table would. In the above examples, the number of data points required in the training and test sets were relatively lower than the number of data entries which would be required to construct a look-up table for the same problem and range of variation. Given that the input spaces of the examples had dimensions varying from 5 (Example A) to 18 (Example D), the size of the tables required for these problems would have been very large, as each dimension of the input space would cause the table size to grow exponentially. For example, in order to construct a look-up table for Example D, which had 18 inputs, even with three uniformly-spaced points along each dimension, a total of 3^{18} data points would be required, which is notably larger than the 9000 points needed to train and test the neural network model.

The run-time memory requirements of simulation using neural network modeling techniques is also significantly lower than using tables or EM simulation techniques. EM simulation programs based on methods such as FEM allocate large amounts of memory during run-time to evaluate structures with large cross sections, and cannot be executed if such machine resources are not available. This limitation is overcome by neural networks, where the memory requirements, given by the sizes of the models, are significantly smaller.

The large number of variables that a neural network is capable of simultaneously handling make neural networks a viable alternative over polynomial curve-fit techniques. In Example C, 30 output variables were mapped simultaneously. In Example D, the number of inputs considered was 18. High-order polynomial curve-fitting with 18 independent input variables would prove to be significantly more difficult than using neural networks. Low-order polynomial models such as quadratic models in 18 variables would have a much lower range of validity than a neural model.

VI. SUMMARY AND CONCLUSIONS

A neural network based approach to the characterization of high-speed interconnects, preserving the accuracy of existing techniques yet simplifying their CPU requirements has

been presented. The general technique used for developing neural network models for interconnects and interconnection networks presented in this paper can be employed to effectively model almost any kind of interconnect configuration or problem one might come across in high-speed VLSI design. The examples discussed show the potential advantages neural networks have over existing techniques, and the CPU resource savings attainable. The low on-line run-time neural network models allow IC designers to perform much more elaborate CAD and optimization, thereby producing better design and yielding results, without expending the large amounts of CPU resources which would be required if existing techniques were used.

ACKNOWLEDGMENT

The authors wish to acknowledge the Dr. G. Wilson of Nortel Technology for access to SALI and the Telecommunications Research Institute of Ontario (TRIO) for their support.

REFERENCES

[1] M. S. Nakhla and Q. J. Zhang, Eds., *Modeling and Simulation of High Speed VSLI Interconnects.* Norwell, MA: Kluwer, 1994.
[2] W. W. M. Dai, Guest Ed., "Special issue on simulation, modeling, and electrical design of high-speed and high-density interconnects," *IEEE Trans. Circuit Syst. I,* vol. 39, pp. 857–982, Nov. 1992.
[3] D. S. Gao, A. T. Yang, and S. M. Kang, "Modeling and simulation of interconnection delays and crosstalk in high-speed integrated circuits," *IEEE Trans. Circuit Syst. I,* vol. 37, pp. 1–9, Jan. 1990.
[4] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design.* New York: Van Nostrand, 1983.
[5] J. W. Bandler and S. H. Chen, "Circuit optimization: The state of the art," *IEEE Trans. Microwave Theory Tech.,* vol. 36, pp. 424–443, Feb. 1988.
[6] K. K. Mihan, B. J. Stacey, M. S. Nakhla, and Q. J. Zhang, "Concurrent thermal and electrical optimization of high-speed packages and systems," in *Int. Intersociety Elec. Packaging Conf.,* Kaanapali Beach, HI, Mar. 1995, pp. 221–227.
[7] *OSA90 Version 2.0 Reference Manual,* Optimization Systems Associates, Inc., Dundas, Ont. Canada.
[8] C. R. Paul, *Analysis of Multiconductor Transmission Lines.* New York: Wiley, 1994.
[9] R. F. Harrington, *Field Computation by Moment Methods.* New York: Macmillan, 1968.
[10] A. R. Djordjevic, R. F. Harrington, and T. K. Sarkar, *Matrix Parameters of Multiconductor Transmission Lines.* Norwood, MA: Artech House, 1989.
[11] J. Poltz, "Optimizing VLSI interconnect model for SPICE simulation," *Int. J. Analog Integrated Circuit Signal Processing,* vol. 5, pp. 87–94, 1994.
[12] G. I. Costache, "Finite element method applied to skin-effect problems in strip transmission lines," *IEEE Trans. Microwave Theory Tech.,* vol. MTT-35, pp. 1009–1013, Nov. 1987.
[13] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in *RC* tree networks," *IEEE Trans. Computer-Aided Design,* vol. CAD-2, pp. 202–211, Mar. 1983.
[14] D. Zhou, S. Su, F. Tsui, D. S. Gao, and J. S. Cong, "A simplified synthesis of transmission lines with a tree structure," *Int. J. Analog Integrated Circuit Signal Processing,* vol. 5, no. 1, pp. 19–30, Jan. 1994.
[15] E. Chiprout and M. S. Nakhla, *Asymptotic Waveform Evaluation and Moment Matching for Interconnect Analysis.* Norwell, MA: Kluwer, 1993.
[16] A. R. Djordjevic, T. K. Sarkar, and R. F. Harrington, "Time-domain response of multiconductor transmission lines," *Proc. IEEE,* vol. 75, pp. 743–764, June 1987.
[17] R. Griffith, E. Chiprout, Q. J. Zhang, and M. S. Nakhla, "A CAD framework for simulation and optimization of high-speed VLSI interconnections," *IEEE Trans. Circuits Syst. I,* vol. 39, pp. 893–906, Nov. 1992.
[18] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design,* vol. 9, pp. 352–366, Apr. 1990.
[19] J. E. Bracken, V. Raghavan, and R. A. Rohrer, "Interconnect simulation with asymptotic waveform evaluation (AWE)," *IEEE Trans. Circuits Syst. I,* vol. 39, pp. 869–878, Nov. 1992.
[20] M. Gilligan and S. Gupta, "A Methodology for estimating interconnect capacitance for signal propagation delay in VLSI's," *Microelectron J,* vol. 26, no. 4, pp. 327–336, May 1995.
[21] U. Choudhury and A. Sangiovanni-Vincentelli, "Automatic generation of analytical models for interconnect capacitances," *IEEE Trans. Computer-Aided Design,* vol. 14, pp. 470–480, Apr. 1995.
[22] EESof Inc., *Microwave SPICE Users' Guide,* Westlake Village, CA, 1988.
[23] A. H. Zaabab, Q. J. Zhang, and M. S. Nakhla, "A neural network approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.,* vol. 43, pp. 1349–1358, June 1995.
[24] B. G. Hoskins and M. R. Haskard, "Artificial neural network techniques for the estimation of thick film resistors," *Microelectron J.,* vol. 26, no. 1, pp. 9–16, Jan. 1995.
[25] P. Ojala, J. Saarinen, P. Elo, and K. Kaski, "Novel technology independent neural network approach on device modeling interface," *Proc. Inst. Elect. Eng.—Circ., Dev. Sys.,* vol. 142, no. 1, pp. 74–82, Feb. 1995.
[26] Q. J. Zhang and M. S. Nakhla, "Signal integrity analysis and optimization of VLSI interconnects using neural network models," in *Proc. IEEE Int. Symp. Circuits Syst.,* London, U.K., May 1994, pp. 459–462.
[27] A. Veluswami, Q. J. Zhang, and M. S. Nakhla, "A neural network model for propagation delays in systems with high speed VLSI interconnect networks," in *Proc. CICC,* Santa Clara, CA, May 1995, pp. 387–390.
[28] A. Veluswami, "Neural network based CAD for high-speed interconnects and monolithic inductors," M.S. thesis, Dept. of Electronics, Carleton Univ., Ottawa, Ont., Canada, Nov. 1995.
[29] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: IEEE Press, 1994.
[30] *Matlab Version 4.2,* The Mathworks, Inc., Natick, MA, 1994.
[31] *SALI: Structure Analysis for Lossy Interconnect,* Northern Telecom Ltd., Ottawa, Ont., Canada, 1995.
[32] R. Griffith and M. S. Nakhla, "Time-domain analysis of lossy coupled transmission lines," *IEEE Trans. Microwave Theory Tech.,* vol. 38, pp. 1480–1487, Oct. 1990.

Anand Veluswami (S'91–M'95) was born on February 8, 1972, in Toronto, Canada. He received the M.S. degree in electrical engineering from Carleton University, Ottawa, Canada, in 1996.

He is currently working as a Software Designer with Enabling Technologies, Mitel Corporation, Kanata, Canada. His research interests are in computer-aided design and simulation of electrical systems, as well as artificial intelligence and their applications to software systems.

Michel S. Nakhla (S'73–M'75–SM'88) received the Ph.D. degree in electrical engineering from the University of Waterloo, Ont., Canada, in 1975.

From 1976 to 1988, he was with Bell-Northern Research, Ottawa, Canada, as the Senior Manager of the computer-aided engineering group. In 1988, he joined Carleton University, Ottawa, Canada, where he is currently a Professor in the Department of Electronics and the Holder of the Computer-Aided Engineering Senior Industrial Chair established by Bell Northern Research and the Natural Sciences and Engineering Research Council of Canada. He is the Founder of the high-speed CAD research group at Carleton University. He serves as Technical Consultant for several industrial organizations. He has authored and coauthored over 100 technical papers and two books on high-speed circuits and interconnects. His research interests include computer-aided design of VLSI and communication systems, high-frequency interconnects, synthesis of analog circuits, wavelets, and neural networks.

Qi-Jun Zhang (S'84–M'85–SM'95) received the B.Eng. degree from the East China Engineering Institute, Nanjing, China, and the Ph.D. degree from McMaster University, Hamilton, Ont., Canada, both in electrical engineering, in 1982 and 1987, respectively.

From 1982 to 1983, he was with the Institute of Systems Engineering, Tianjin University, Tianjin, China. From 1988 to 1990 he was a Research Engineer with Optimization Systems Associates, Inc., Dundas, Ont., Canada. During 1989 and 1990, he was also an Assistant Professor (part-time) of electrical and computer engineering at McMaster University. In 1990, he joined the Department of Electronics, Carleton University, Ottawa, Canada, where he is presently an Associate Professor. He is a coeditor of *Modeling and Simulation of High-Speed VLSI Interconnects* (Kluwer, 1994) and a contributor to *Analog Methods for Computer-Aided Analysis and Diagnosis* (Marcel Dekker, 1988). His professional interests include all aspects of circuit CAD with emphasis on large scale simulation and optimization, statistical design and modeling, neural networks, sensitivity analysis, and optimization of microwave circuits, and high-speed VLSI interconnections.