In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.

While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.

Adjoint Dynamic Neural Network Technique for Accurate Sensitivities in Nonlinear Transient Modeling and High-Speed Interconnect Design

by

Yi Cao, B. Eng.,

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirement for the
degree of Master of Applied Science

Ottawa-Carleton Institute for Electrical and Computer Engineering
Department of Electronics
Carleton University
Ottawa, Ontario K1S 5B6
Canada

©Copyright April 2003, Yi Cao

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file   Votre référence

ISBN: 0-612-88867-3

Our file   Notre référence

ISBN: 0-612-88867-3

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Canada

Abstract

In this thesis, we propose a new adjoint dynamic neural network (ADNN) technique aimed at enhancing computer-aided design (CAD) of high-speed VLSI modules. A novel formulation for exact sensitivities is derived employing Lagrange functions approach, and by defining an adjoint of a dynamic neural network (DNN), for the first time. The proposed ADNN is a dynamic model that we solve using integration backwards through time. A single ADNN solution can be used to efficiently compute exact sensitivities of the corresponding DNN with respect to all its parameters. Using these sensitivities, we develop a training algorithm that facilitates DNN learning of nonlinear transients directly from continuous time-domain waveform data. Resulting accurate and fast DNN models can be straightaway used for carrying out high-speed VLSI CAD in SPICE-like time-domain environment. The technique can also speed-up physics-based nonlinear circuit CAD through faster sensitivity computations. Applications of the proposed ADNN technique in transient modeling and nonlinear design are demonstrated through several high-speed interconnect driver modeling and circuit optimization examples.

Acknowledgements

First of all, I would like to express my deepest gratitude to my supervisor, Prof. Qijun Zhang, for his professional guidance, invaluable inspiration, continuous financial support and patience throughout the research work and the preparation of this thesis. His constant striving for rigorous research and profound knowledge will influence and benefit me for the rest of my life.

I am also grateful to Betty Zahalan, Peggy Piccolo, Nagui Mikhail, Jacques Lemieux and all other staff and faculty for providing the great lab facilities and enjoyable environment for study and research.

Xiaolei Ding, Vijay K. Devabhaktuni, Lei Zhang and other students in my group are thanked for the help, friendship and useful discussions.

My special thank is given to Prof. Mustapha C.E. Yagoub and Jianjun Xu, who have generously given me numerous technical assistance throughout my research work.

Finally, I would like to thank my parents. Without their love, support and encourage, the accomplishment of this thesis would not have been possible.

Contents

Chapter 1 Introduction	1
1.1 Thesis Motivation	1
1.2 Thesis Objective	3
1.3 Thesis Organization	4
 Chapter 2 Literature Review	 6
2.1 Artificial Neural Networks for RF/Microwave Circuit Design	6
2.2 Neural Network Structures	10
2.2.1 Multilayer Perceptron Networks	10
2.2.2 Knowledge Based Neural Networks	13
2.2.3 Neural Networks with Feedbacks	17
2.3 Neural Network Training Algorithms	23
2.3.1 Training Objective	23
2.3.2 Back Propagation Algorithm	24
2.3.3 Gradient-Based Optimization Methods	27
2.3.4 Global Optimization Methods	31
2.4 Review of Adjoint Sensitivity Techniques.....	32
2.4.1 The Fundamentals of Adjoint Network and Sensitivities	33
2.4.2 Important Applications of Adjoint Techniques	37

2.5	Conclusions.....	40
Chapter 3	Proposed Adjoint Dynamic Neural Network Technique	42
3.1	Introduction	42
3.2	Dynamic Neural Network	45
3.3	Proposed ADNN Sensitivity Technique	46
3.3.1	Problem Statement.....	46
3.3.2	Proposed ADNN	47
3.3.3	Calculations of Derivatives	49
3.4	ADNN Sensitivities for DNN Training	50
3.4.1	Training Objective	50
3.4.2	Further Theoretical Relations between ADNN and DNN	53
3.4.3	Algorithm of ADNN-Enabled Transient Training	56
3.4.4	ADNN Based Algorithm for Determining Sensitivity w.r.t. Static Inputs to DNN	59
3.5	Summary	59
Chapter 4	Application Examples of Proposed ADNN Technique	62
4.1	CMOS Inverter	63
4.2	Circuit-Based High-Speed Interconnect Driver	67
4.3	Physics-Based Multistage Driver	71
4.4	Physics-Based Interconnect Driver Optimization and Design	77

Chapter 5 Conclusions and Future Research	87
5.1 Conclusions	87
5.2 Suggestions for Future Research	88
Bibliography	91

List of Figures

Figure 2.1: Multilayer Perceptrons (MLP) neural network structure. Typically, a MLP network consists of one input layer, one or more hidden layers, and one output layer	12
Figure 2.2: Knowledge based neural network (KBNN) structure. Typically a KBNN consists of six layers.	16
Figure 2.3: The discrete RNN based macromodel structure for nonlinear dynamic modeling. It includes 3-layer MLP structure and feedback loop.	18
Figure 2.4: <i>Initial training</i> : to train the f_{ANN} part in time-domain using spectrum data, where $A^{(i)}$ is the time derivative of Fourier Transform matrix.	22
Figure 2.5: Circuit representation of the DNN model for incorporation into high-level simulation.	23
Figure 3.1: Block diagram of the ADNN enabled training algorithm used to train DNN to learn transient waveform data. D is an operator for time-differentiation. Both the original DNN and the proposed ADNN are evaluated using numerical integration in each epoch of training, providing the information of training error and its derivatives w.r.t. DNN internal weights w	52

Figure 3.2: Block schematic of the ADNN algorithm for sensitivity analysis w.r.t. DNN static inputs s . This technique is proposed for circuit design purpose using trained DNN. The DNN is excited by the circuit signal $u(t)$ and evaluated by numerical integration forward in time. The proposed ADNN is evaluated using numerical integration backward in time. The excitation signal to the ADNN is derived from the solutions of the original DNN and the energy function. 61

Figure 4.1: Schematic of CMOS inverter circuit of example 4.1 and its corresponding electric variables. Training waveforms of the inverter are generated using level-49 BSIM3V3 *HSPICE* MOSFET model 64

Figure 4.2: Sensitivity of DNN training error w.r.t. different weight parameters in DNN for the CMOS inverter in example 4.1. (a) Sensitivity w.r.t. weights in the output layer of f_{ANN} , and (b) Sensitivity w.r.t. weights in the hidden layer of f_{ANN} . Excellent agreement is observed between the proposed ADNN technique and the perturbation method. 66

Figure 4.3: A coupled 3-conductor interconnect circuit terminated with nonlinear buffers. This circuit is used to generate testing waveforms for testing trained DNN models of nonlinear drivers. Varying parameters in the pulse signal generator can allow us to obtain different sets of testing waveforms..... 69

Figure 4.4: (a) Signal propagation waveforms v_{out1} and (b) crosstalks waveforms v_{out2} and v_{out3} of the VLSI interconnect example of Figure 4.3. Excellent agreement is achieved between the responses of DNN trained using the proposed ADNN

technique, with those obtained from original HSPICE simulation.	70
Figure 4.5: A 4-stage CMOS driver circuit used in example 4.3. DNN models are trained using proposed ADNN technique to learn the transient input-output waveform data generated using physics-based semiconductor simulator such as <i>MINIMOS-NT</i>	72
Figure 4.6: The output voltage waveforms of the 4-stage CMOS driver under different excitations and driving different transmission line loads. Excellent agreement is achieved between the responses of DNN trained using proposed ADNN technique, with those of <i>MINIMOS-NT</i> . The fact that these <i>MINIMOS-NT</i> test waveforms are never used in training confirms the validity of DNN model.	73
Figure 4.7: Monte-Carlo analysis using the DNN model of physics-based high-speed driver for signal integrity. The fast and accurate DNN model is developed using proposed ADNN-enabled training algorithm.	74
Figure 4.8: The sensitivities of the average output power of the 4-stage CMOS driver evaluated at 3 different driver sizes. The average power was used as the energy function E while computing the sensitivities using the proposed ADNN technique.....	76
Figure 4.9: The output voltage waveforms of the 4-stage CMOS driver for 3 different geometries: (1) $W_1=1.0\mu\text{m}$ $S=2.5$; (2) $W_1=1.5\mu\text{m}$ $S=4.0$; (3) $W_1=3.0\mu\text{m}$ $S=3.0$. Excellent agreement is achieved between the responses of DNN with	

those of *MINIMOS-NT* even though these *MINIMOS-NT* test waveforms are never used in training. 78

Figure 4.10: Sensitivity of the objective function for time-domain signal integrity optimization of high-speed VLSI interconnects (in example 4.4) w.r.t. scaling coefficient S for a subset of W_1 & S parameters used for training (1) $W_1=1.0\mu\text{m}$; (2) $W_1=1.3\mu\text{m}$ and (3); $W_1=1.6\mu\text{m}$. The one-sided least p^{th} function was used to formulate the energy function E while computing the sensitivities using the proposed ADNN technique. 83

Figure 4.11: Sensitivity of the objective function for time-domain signal integrity optimization of high-speed VLSI interconnects (in example 4.4) w.r.t. 1st stage buffer width for a subset of W_1 & S parameters used for training (1) $S=2.0$; (2) $S=2.1$; and (3) $S=2.2$. The one-sided least p^{th} functions was used to formulate the energy function E while computing the sensitivities using the proposed ADNN technique. 84

Figure 4.12: (a) Responses of the 4-stage CMOS driver models before and after signal integrity optimization. The DNN based optimal solutions are verified by direct physics-based *MINIMOS-NT* simulation. (b) Example of ADNN inputs and outputs used to generate sensitivities for optimization. DNN based optimization utilizing proposed ADNN sensitivity achieves physics-level accuracies at only a tiny fraction of the CPU time required by direct physics based optimization..... 85

Figure 4.13: (a) Responses of the 4-stage CMOS driver models before and after signal integrity optimization. The DNN based optimal solutions are verified by direct physics-based *MINIMOS-NT* simulation. (b) Example of ADNN inputs and outputs used to generate sensitivities for optimization. Compared with Figure 4.12 these results are obtained for long transmission line loads, yielding optimal buffer sizing for different interconnects.

..... 86

List of Tables

Table 2-I: The summaries of the adjoint network branch relations and sensitivity coefficients	36
Table 4-I: Comparison of sensitivity values of CMOS inverter DNN model in example 4.1	65
Table 4-II: DNN driver model accuracies for different cases in example 4.2	68
Table 4-III: Time for training DNN driver model in example 4.2	68

Chapter 1

Introduction

1.1 Thesis Motivation

In recent years, radio frequency (RF) and microwave circuits are experiencing rapid growth and development on both the research and industrial sides leading to shrinking device dimensions and increased complexities of circuit functionality. The advanced computer-aided design (CAD) tools that allow extensive design, analysis and optimization are highly demanded for designing such high frequency electronic circuits. By effectively using CAD tools, the behavior of the large-scale circuit and system such as the performance, manufacturability and electrical and physical reliability could be predicted and optimized before any physical hardware prototyping.

For the modern complex circuits and systems, the high integration level of nonlinear elements would cause long run-time and require intensive computation resources. In addition, currently employed CAD tools offer high-level optimization based on highly repetitive routines. Statistical analysis and yield optimization are such examples, which require solving the complicated nonlinear circuit iteratively and yield more intensive computations.

For the nonlinear circuit, its property could be best represented in the continuous time-domain. An interesting example of large-scale nonlinear circuit is the interconnect network with nonlinear terminations. Typically, in the state of the art of VLSI circuit, there could exist tremendous such topologies. In general, physical/geometric sizes of termination buffers could be very useful in the actual optimization and signal integrity of such nonlinear circuits. How to efficiently model the transient behaviors of nonlinear devices including such interconnect network under different physical /geometric parameters of nonlinear buffers is the primary motivation of the thesis.

Artificial neural networks (ANN) are powerful information-processing systems, which possess the distinguished ability of learning and abstracting from complex relationship existed in the input-output information [1]. Over the decades, it has been successfully applied in the various fields of physical science and engineering. Since the ANN concept was firstly introduced to the area of RF/Microwave design in 1993, it has been recognized as a fast and accurate vehicle for microwave modeling, optimization and design [1,2]. The ANN-based modeling techniques contain all the desired features of ideal modeling techniques. The model is developed through a computer-driven process thus avoiding manually creating equivalent circuit topology. Meanwhile, it can easily handle multi-dimensional problem with less memory requirement compared to table look-up method. In theory, the universal approximation theorem would ensure that the developed neural models accurately represent arbitrary continuous nonlinear relations between inputs and outputs [1]. On the other hand, the property of its fast evaluation speed allows neural models to be substantially used in on-line highly repetitive design stage such as yield optimization and statistic design.

Recently, as nonlinear microwave circuits target more shirked device sizes and complex functionaries, nonlinear dynamic CAD has become one of the most important applications of ANNs [3,4]. The existing ANN-based approaches have been focused on large-signal modeling and design in the case of nonlinear periodic RF/microwave responses in harmonic balance (HB) environment. To further demonstrate the advantages of neural networks for nonlinear transient modeling and high-speed interconnect design is another crucial motivation of the thesis.

1.2 Thesis Objective

The objective of the thesis is to develop a systematic sensitivity technique to enable efficient nonlinear transient modeling through DNN and high-speed VLSI interconnect design. The significant contributions of this work are as follows:

- For the first time, a new ADNN technique [5] is proposed for nonlinear transient scenarios such as those in high-speed interconnects with nonlinear terminations. An elegant formulation for exact adjoint sensitivities is derived employing Lagrange functions approach, by defining an adjoint of the DNN.
- Based on the proposed ADNN technique, a robust training algorithm that facilitates efficient DNN training directly from transient waveform data is developed. The trained model is in continuous time-domain format which best describes the fundamental essence of nonlinear circuits.
- Sensitivities of DNN energy function with respect to DNN static inputs are also derived. These sensitivities can lead to faster nonlinear circuit design including the effect of geometrical/physical parameters. With the proposed

approach, the physics-based high-speed interconnect design becomes practically feasible yielding enhanced design accuracy.

1.3 Thesis Organization

The rest of the thesis is presented in the following way.

Chapter 2 gives a conceptual overview of neural network approach for microwave/RF modeling and design. Firstly the-state-of-the-art of ANN applications in RF/Microwave area is described. Then different important neural network structures including both static and dynamic networks are reviewed. As training is the most important step in model development, some widely applied training algorithms are also presented. In the end of this chapter, some existing adjoint sensitivity techniques for circuit-based, EM-based and ANN-based applications are described as well.

In Chapter 3, the proposed ADNN sensitivity technique is discussed in details. To begin with, we give a brief description of the DNN formulation for modeling nonlinear dynamic circuits. Following that, the proposed ADNN is presented and the exact sensitivities of the energy function for DNN system with respect to both DNN internal weights and external inputs are systematically derived. As a direct application of the resulting sensitivity formulae, we show how the dynamic training error is related to the energy function and describe a new training algorithm for DNN to learn transient data.

Chapter 4 presents several examples to illustrate the validity and efficiency of the proposed ADNN technique. DNN transient modeling examples of both circuit-based and physics-based high-speed drivers are presented. To show the relevance of proposed ADNN technique for nonlinear transient design, an example of physics-based driver

circuit optimization is also included. The advantages of using the ADNN are demonstrated through substantial speedup in transient based training of the buffer models, and signal integrity based optimization.

Finally, Chapter 5 contains the conclusion of the thesis and some possible directions for future research.

Chapter 2

Literature Review

2.1 Artificial Neural Networks for RF/Microwave

Circuit Design

The rapid growth of telecommunication and wireless industry in the recent years has resulted in a drive for continuous improvement of the performance of Radio Frequency (RF) and microwave circuits. The wireless industry's emphases on time-to-market and low manufacturing cost, resulting from first-pass-design, are putting the demands on reliability and efficiency of CAD tools for RF/microwave circuit design. New modeling techniques with high accuracy and fast model evaluation are highly required for circuit optimization and yield-driven design.

Neural networks, also called Artificial Neural Networks (ANN), are information processing systems with their design inspired by the studies of ability of human-brain to learn from observations and to generalize by abstraction [1]. The fact that neural networks can be trained to learn any arbitrary nonlinear input-output relationships from corresponding data has resulted in their use in a number of areas such as pattern recognition, control, speech processing etc. Over the last decade, a CAD approach based

on ANN gained recognition in RF, microwave and VLSI community [1,2,6]. Fast, accurate, and reliable neural network models (or simply neural models) can be directly developed from measured/simulated microwave data. Once developed, these neural models can be used to replace computationally intensive physics/EM models of active/passive devices to speed up microwave design [1][2]. Through the intensive researches, neural network techniques have been demonstrated to model a wide variety of microwave devices/circuits such as transmission line components [7-20], bends [21, 22], vias [23-28], CPW components [26, 28, 29], spiral inductors [30, 31], microstrip component [12,32], FETs [4, 7, 8, 13, 15, 19, 33-38], HBT [35, 39, 40], HEMT [41, 42], waveguides [26, 43], amplifiers [3-4,15, 35, 37, 44-47], mixers [3,4], antennas [29, 48-54]. Neural networks have also been used in object wave propagation [55, 56], impedance matching [57-59], inverse modeling [43, 60-62], circuit design and optimization [11, 12, 14, 15, 18, 22, 28, 35, 37, 44, 58, 61, 63-64], synthesis [45, 64], and yield optimization [44]. Neural models are much faster than original detailed physics/EM models, more accurate than polynomial and empirical models, allow more dimensions than table lookup models, and are easier to develop when a new device/technology is introduced [11].

To develop neural models, it involves many crucial and nontrivial issues such as data generation, data scaling, selecting neural network structure, neural network training etc. [1]. Two most important issues in the neural development are the determination of suitable neural network structure and the training of constructed neural network.

The structure issue is closely related to the property of modeling problems. Microwave modeling problems could be broadly divided into two categories, i.e., static case and dynamic case. Many neural networks are capable of modeling nonlinear, multidimensional problems, but different neural network structures may differ in efficiency and accuracy when representing engineering problems. In fact, appropriate structure may give better accuracy and require less training data and computation expense in training. If the input and output relationship of the problem is algebraic, various feed forward neural networks can be adopted. The most widely used neural network structure is Multilayer Perceptron (MLP). However, in terms of training data and computation expense, a MLP network may use tens of hidden neurons and volumes of training data in predicting a nonlinear circuit waveform while for knowledge based neural networks, in which existing knowledge is embedded with the some particular neurons, less training data and hidden neurons are required as well as higher accuracy can be achieved [7]. Recently, nonlinear dynamic modeling has gained attention for the CAD of large-scale RF/ Microwave circuits. In order to characterize the dynamic analog behavior of nonlinear circuits, where time domain information needs to be used, conventional feed forward neural networks are not sufficient. To address this, both discrete Recurrent Neural Network (RNN) and Dynamic Neural Network (DNN) approaches have been reported in the literature [3][4].

Another aspect of the structure issue is to determine the appropriate number of hidden neurons, which represents the ability of neural network to capture problem nonlinearity. Theoretically, the more nonlinear the problem is, the more hidden neurons are needed.

But too many hidden neurons will lead to overlearning, and too few hidden neurons will not represent the problem sufficiently, i.e. underlearning [1].

To begin with, a neural network does not know any information about the microwave device/circuit problem. In order to represent the device/circuit behavior, the neural network has to be trained with the corresponding data. As such, neural network training is the most crucial step in model development. A good collection of the training data, which is well distributed, sufficient, and accurately measured/simulated, is the basic requirement to obtain an accurate model. Beyond that, choice of suitable training algorithms also plays a very important role in the success of model development. A good training algorithm will speed up the training procedure with better accuracy. Back Propagation (BP) [65] scheme was introduced to provide the derivative of output error w.r.t. each weight in the MLP network in mid 1980's. Based on this scheme, basic weight updating and its variation methods were proposed firstly. After that, second-order gradient-based optimization methods have been applied because the training process is essentially an optimization procedure. Recently, global optimization techniques such as Simulated Annealing [66] and Genetic Algorithm [67] were proposed to overcome the problem of local minimum in the gradient-based training methods.

In the following two sections, several popular neural network structures are first reviewed according to their functionalities and applications. Then both gradient-based and global training algorithms are presented.

2.2 Neural Network Structures

2.2.1 Multilayer Perceptron Networks

Multilayer Perceptron (MLP) network is the most popularly used and basic type of neural networks that has been used in most of the microwave applications [1,2]. In MLP structure, the neurons are grouped into layers. The neurons in the same layer have no connection to each other and the neurons between consecutive layers are fully connected. Typically, an MLP neural network consists of an input layer, one or more hidden layers, and an output layer, as shown in Figure 2.1. Suppose the total number of layers is L . The first layer is the input layer, the L^{th} layer is the output layer, and layers 2 to $L-1$ are hidden layers. Suppose the number of neurons in l^{th} layer is N_l , $l = 1, \dots, L$. Let w_{ij}^l represent the weight of the link between j^{th} neuron of $(l-1)^{th}$ hidden layer and i^{th} neuron of l^{th} hidden layer, w_{i0}^l and z_i^l be the bias parameter and the output of i^{th} neuron of l^{th} hidden layer. So the vector of all trainable weights in a MLP is

$$\Phi = [w_{10}^2 \quad w_{11}^2 \quad w_{12}^2 \quad \dots \quad w_{N_L N_{L-1}}^L]^T \quad (2.1)$$

Let x_i represent the i^{th} input parameter to the MLP. Then following equations stand:

$$z_i^1 = x_i, \quad i = 1, \dots, N_1 \quad (2.2)$$

$$z_i^l = \sigma \left(\sum_{j=1}^{N_{l-1}} w_{ij}^l z_j^{l-1} + w_{i0}^l \right), \quad i = 1, \dots, N_l, \quad l = 2, \dots, L-1 \quad (2.3)$$

where $\sigma(\cdot)$ is the activation function of hidden neurons. The most commonly used form of $\sigma(\cdot)$ is the logistic sigmoid function given by

$$\sigma(\gamma) = \frac{1}{(1 + e^{-\gamma})} \quad (2.4)$$

It is a smooth switch from 0 to 1 as γ varies from negative infinity to positive infinity.

Other possible candidates for $\sigma(\cdot)$ could be arctangent function

$$\sigma(\gamma) = \left(\frac{2}{\pi}\right) \arctan(\gamma) \quad (2.5)$$

or hyperbolic tangent function

$$\sigma(\gamma) = \frac{(e^\gamma - e^{-\gamma})}{(e^\gamma + e^{-\gamma})} \quad (2.6)$$

All of them are bounded, continuous, monotonic and continuously differentiable.

The outputs of MLP are produced by linear functions in the output neurons, and can be computed as

$$y_k = \sum_{i=1}^{N_{L-1}} w_{ki} z_i^{L-1} + w_{k0}^L, \quad k = 1, \dots, N_L \quad (2.7)$$

The universal approximation theorem of MLP was proved by Cybenko [68] and Hornik et al. [69] in 1989. According to the universal approximation theorem, provided sufficient hidden neurons, a 3-layer perceptron network is theoretically capable of approximating an arbitrary continuous, multi-dimensional real static nonlinear function to any desired accuracy. But in practice, how to choose the number of hidden neurons is still an open question. A definite answer is that, the number of hidden neurons depends on the degree of nonlinearity and the dimension of the original problem. More nonlinear and more dimensional problems need more hidden neurons. But too many hidden neurons will result in more redundant freedom in Φ , thus lead to the problem of overlearning [1]. Several algorithms have been proposed to find a proper size of MLP network such as constructive algorithm [70] and network pruning [71].

Figure 2.1: Multilayer Perceptrons (MLP) neural network structure. Typically, a MLP network consists of an input layer, one or more hidden layers, and an output layer.

In practice, 3-layer or 4-layer perceptron networks are commonly used for RF/Microwave applications. Intuitively, a 4-layer perceptron network would perform better for tasks that have certain common localized behavior in different regions of the problem space. For the same task, a 3-layer perceptron network however requires lots of hidden neurons to repeat the same behavior in different input regions. In [72], it is pointed out that 3-layer perceptron networks are preferred in function approximation where generalization capability is a major concern. In [73], the author demonstrated that 4-layer perceptron networks would have better performance in boundary definition and are favored for pattern classification tasks.

2.2.2 Knowledge Based Neural Networks

In the neural networks described in last section, no problem dependant information/knowledge appears in the structure. Therefore, microwave modeling using MLP neural networks is often called purely black box modeling. Black box modeling is easy to implement, however it requires a tremendous amount of training data, which is often very expensive to generate for microwave applications [1]. To combine the abundant existing microwave knowledge with neural network modeling technique, Knowledge Based Neural Network (KBNN) was presented for RF/Microwave oriented modeling and design aimed at enhancing the neural model accuracy and reducing the need for massive training data [7]. When more knowledge exists, the KBNN model can be developed with less training data and time as well as better extrapolation capability. On the contrary, if only less knowledge is available then more training data has to be used to ensure the model accuracy.

The structure of KBNN is shown in Figure 2.2. The microwave knowledge in the form of empirical formula is inserted as a part of the overall neural network structure. There are 6 layers in the structure that are not fully connected to each other. Namely, these 6 layers are input layer X , knowledge layer K , boundary layer B , region layer R , normalized layer $\bar{R}$ and output layer Y . The input layer X accepts parameters $\mathbf{x}$ from outside of the model. The knowledge layer K is the place where existing knowledge resides in the form of single or multidimensional functions $\psi(\cdot)$. Output of knowledge neuron i in the K layer is given by [7]

$$k_i = \psi_i(\mathbf{x}, \mathbf{w}_i), \quad i = 1, \dots, N_k \quad (2.8)$$

where $\mathbf{x}$ is the vector including neural network inputs x_i ($i = 1, \dots, N_x$), N_k is the number of knowledge neurons, and $\mathbf{w}_i$ is the vector of all the parameters in the knowledge formula. The knowledge function $\psi_i(\mathbf{x}, \mathbf{w}_i)$ is usually in the form of empirical or semi-analytical functions. For example, the mutual inductance of a transmission line is a function of conductor width, separation between conductors and height of substrate [74]. The boundary layer B can incorporate knowledge in the form of problem dependent boundary functions $B(\cdot)$; or in the absence of boundary knowledge just as linear boundaries. Output of the i^{th} neuron in this layer b_i is described by

$$b_i = B_i(\mathbf{x}, \mathbf{v}_i), \quad i = 1, \dots, N_b \quad (2.9)$$

where $\mathbf{v}_i$ is a vector of all parameters in B_i representing an open or closed boundary in the input space $\mathbf{x}$, and N_b is the number of boundary neurons [7]. The region layer R contains neurons to construct regions from boundary neurons, whose outputs $\mathbf{r}$ can be computed as

$$r_i = \prod_{j=1}^{N_b} \sigma(\alpha_{ij} b_j + \theta_{ij}), \quad i = 1, \dots, N_r \quad (2.10)$$

where α_{ij} and θ_{ij} are the scaling and bias parameters, respectively, and N_r is the number of region neurons. Here $\sigma(\cdot)$ denotes a sigmoid function [1]. The normalized region layer $\bar{R}$ contains rational function based neurons [75] to normalize the outputs of region layer,

$$\bar{r}_i = \frac{r_i}{\sum_{j=1}^{N_r} r_j}, \quad i=1, \dots, N_{\bar{r}}, \quad \text{where } N_{\bar{r}}=N_r \quad (2.11)$$

The output layer Y contains second order neurons combining the outputs from both knowledge neurons and normalized region neurons [7]

$$y_j = \sum_{i=1}^{N_k} \beta_{ji} k_i \left(\sum_{s=1}^{N_{\bar{r}}} \rho_{jis} \bar{r}_s \right) + \beta_{j0}, \quad j = 1, \dots, N_y \quad (2.12)$$

where β_{ji} reflects the contribution of the i^{th} knowledge neuron to output neuron y_j and β_{j0} is the bias parameter. ρ_{jis} is a trainable region selector. If it is 1 it means that region $\bar{r}_s$ is the effective region of the i^{th} knowledge neuron contributing to the j^{th} output. A total of $N_{\bar{r}}$ regions are shared by all the output neurons. As a special case, if we assume that each normalized region neuron selects a unique knowledge neuron for each output j , the function for output neurons can be simplified as

$$y_j = \sum_{i=1}^{N_k} \beta_{ji} k_i \bar{r}_i + \beta_{j0}, \quad j = 1, \dots, N_y \quad (2.13)$$

Training parameters ψ for the entire KBNN model includes [7]

$$\psi = [\mathbf{w}_i \ i = 1, \dots, N_k; \ \mathbf{v}_i \ i = 1, \dots, N_b; \ \alpha_{ij}, \theta_{ij} \ i = 1, \dots, N_r, \ j = 1, \dots, N_b; \ \beta_{ij} \ j = 1, \dots, N_y, \ i = 0, \dots, N_k; \ \rho_{jik} \ j = 1, \dots, N_y, \ i = 1, \dots, N_k, \ k = 1, \dots, N_{\bar{r}}] \quad (2.14)$$

Compared to pure black box neural network structures such as MLP, the prior knowledge in KBNN gives neural network more information about the original microwave problem, beyond what is already included in the training data. Consequently, KBNN models have better reliability when training data is insufficient or when the model is used outside the model training range.

Figure 2.2: Knowledge Based Neural Network (KBNN) structure. Typically a KBNN consists of six layers.

2.2.3 Neural Networks with Feedbacks

A. Recurrent Neural Network

Conventional feed-forward neural networks are well known for their learning and generalization capabilities. However, they can only map input-output relationships statically. To model nonlinear circuit responses in time-domain, a neural network that can include temporal information is necessary. Recurrent neural network (RNN) has been found to be a suitable candidate to accomplish this job. In the past, RNNs were successfully used in engineering applications such as system control [76, 77], speech recognition [78] etc.

For microwave dynamic modeling, the structure of a typical RNN is shown in Figure 2.3 [3]. The input layer includes time-varying inputs u and time-invariant inputs p . The neural network outputs are the time varying signal y . The first hidden layer of RNN denoted by x contains buffered (time-delayed) history of y fed back from the output layer, buffered history of u , and p . The second hidden layer z contains neurons with sigmoid activation functions. The last layer of RNN structure is the output layer called y layer. The y layer outputs are linear functions of the responses of hidden neurons in the z -hidden layer. Let the trainable parameters of the RNN be denoted as Φ . As we can observe from the Figure 2.3, the overall RNN structure, including both the feed forward part and feedback connections, realizes the following nonlinear dynamic relationship [3]

$$y(k) = f(y(k-1), \dots, y(k-K_y), u(k-1), \dots, u(k-K_u), p, \Phi) \quad (2.15)$$

Here the number of delay buffers K_y and K_u represent the order of dynamics in the RNN.

Figure 2.3: The discrete RNN based macromodel structure for nonlinear dynamic modeling. It includes 3-layer MLP structure and feedback loop.

B. Dynamic Neural Network

The aforementioned RNN is in a discrete format where the model dynamics are described by a set of difference equations. Thus the developed neural model is not possible for incorporation into typical nonlinear circuit simulator. For the purpose of circuit simulation, the most ideal format to describe nonlinear dynamics is the continuous-time domain. In theory this format best describes the fundamental essence of nonlinear behavior and in practice it is flexible to fit most or all the needs for nonlinear circuit simulation. To fulfill this purpose, dynamic neural network (DNN) [4] technique was presented for large signal modeling of nonlinear microwave circuits and systems.

In time domain, the original nonlinear circuit can be generally described in state equation form as,

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \boldsymbol{\varphi}(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= \boldsymbol{\psi}(\mathbf{x}(t), \mathbf{u}(t))\end{aligned}\tag{2.16}$$

where $\mathbf{u}$ and $\mathbf{y}$ are vectors of the input and the output signals of the nonlinear circuit, respectively. Here $\mathbf{x}$ is a vector containing state variables, $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ represent nonlinear functions. In a modified nodal formulation [79], the state vector $\mathbf{x}$ includes nodal voltages, currents of inductors, currents of voltage sources and charge of nonlinear capacitors.

At system level, such detailed state equations are too large, computationally expensive, and sometimes even unavailable at system level. Therefore, a simpler (reduced order) model approximating the same dynamic input-output relationships is highly required. Let n be the order of DNN representing the reduced order for original nonlinear circuit. Let $\mathbf{v}_i$ be a N_y -vector, $i = 1, 2, \dots, n$. Let $\mathbf{f}_{ANN}$ represent a multilayer

perceptron neural network [1,2] with input neurons representing y , u , their derivatives $d^i y/dt^i$, $i=1, 2, \dots, n-1$, and $d^k u/dt^k$, $k=1,2, \dots, n$; and the output neuron representing $d^n y/dt^n$. In [4], DNN model was formulated as

$$\begin{aligned} \dot{v}_1(t) &= v_2(t) \\ &\vdots \\ \dot{v}_{n-1}(t) &= v_n(t) \\ \dot{v}_n(t) &= f_{ANN}(v_n(t), v_{n-1}(t), \dots, v_1(t), u^{(n)}(t), u^{(n-1)}(t), \dots, u(t)) \end{aligned} \quad (2.17)$$

where inputs and outputs of the DNN model are $u(t)$ and $y(t) = v_1(t)$, respectively.

The overall DNN model (2.17) is in a standardized format for typical nonlinear circuit simulators. For example, the left-hand-side of the equation provides the charge (Q) or the capacitor part, and the right-hand-side provides the current (I) part, which are the standard representation of nonlinear components in many harmonic balance (HB) simulators [80]. DNN can provide dynamic current-charge parameters for general nonlinear circuits with any number of internal nodes in original circuit. In this way, DNN model outperforms the previous static I-Q neural model of [44] which was only suitable for intrinsic FETs.

The DNN order (or the number of hidden neurons) represents the effective order (or the degree of nonlinearity) of the original circuit that is visible from the input-output data [4]. Therefore the size of the DNN reflects the internal property of the original circuit rather than external signals, and as such the model does not suffer from curse of dimensionality in multi-tone simulation. On the other hand, by adjusting DNN order and number of hidden neurons, the DNN model creation becomes much easier than conventional equivalent circuit based approaches [81] where manual trial-and-error may

be needed to create/adjust the equivalent circuit topology and the nonlinear equation terms in it.

DNN model would represent a nonlinear microwave circuit only after we train it with data from the original circuit. Let $\bar{U}(\omega)$ and $\bar{Y}(\omega)$ be such input and output spectrums respectively, $\omega \in \Omega$, where Ω is the set of spectrum frequencies. Let matrix $A(\omega, t)$ be the coefficients of Inverse Fourier Transform [82]. The training process is divided into two stages, i.e. initial training and final training. The process of initial training is illustrated in Figure 2.4 [4]. This process is computationally efficient and can train the f_{ANN} from a random start to an approximate solution. Using results from initial training as starting point, final training is performed in frequency-domain involving harmonic balance (HB) solutions of the DNN model. After training is finished, the exact circuit representation of DNN model, as shown in Figure 2.5 [4], makes it feasible to use the trained DNN model in online circuit simulation. The DNN technique allows further realizing the flexibility of neural based approaches in nonlinear microwave modeling, simulation and optimization.

Figure 2.4: *Initial training*: to train the f_{ANN} part in time-domain using spectrum data, where $A^{(i)}$ is the time derivative of Fourier Transform matrix.

Figure 2.5: Circuit representation of the DNN model for incorporation into high-level simulation.

2.3 Neural Network Training Algorithms

2.3.1 Training Objective

A neural network in the beginning will not represent any device/circuit behavior unless it's trained with corresponding device/circuit data. A neural network model can be developed through an optimization process called training. For microwave modeling purposes, the development of neural models is essentially a curve-fitting problem in a multi-dimensional input space [1]. The training process is accordingly equivalent to

finding proper surfaces in the input space that best fit the training data. Suppose the training data set has N_d sample pairs, $\{(\mathbf{x}_i, \mathbf{d}_i), i=1,2,\dots,N_d\}$, where $\mathbf{x}_i$ and $\mathbf{d}_i$ are N_x - and N_y - vectors representing the inputs and outputs of the training data respectively.

Let $\mathbf{y} = \mathbf{y}(\mathbf{x}, \Phi)$ represent the input-output relationship of the neural network. The objective of training is to find variables Φ such that the cost function $E(\Phi)$ of the error between the neural network predictions and the training data is minimized,

$$\min_{\Phi} E(\Phi) = \sum_{i=1}^{N_d} e_i(\Phi) = \sum_{i=1}^{N_d} (\mathbf{y}_i(\Phi) - \mathbf{d}_i)^2 \quad (2.18)$$

where $e_i(\Phi)$ is the error value of i^{th} sample and $\mathbf{y}_i(\Phi)$ is the output of neural network under inputs $\mathbf{x}_i$. The cost function $E(\Phi)$ is a nonlinear function with respect to the adjustable parameter Φ . Due to the complexity of $E(\Phi)$, iterative algorithms are often used to explore the parameter space efficiently. In iterative descent methods [83], the next point of Φ , denoted as Φ_{next} , is updated by a step down from the current Φ_{now} along with a direction vector $\mathbf{h}$,

$$\Phi_{next} = \Phi_{now} + \eta \mathbf{h} \quad (2.19)$$

where η is a positive step size regulating the extent to which we can proceed in that direction. As we will see in the next several sections, every training algorithm has its own way for weights update.

2.3.2 Back Propagation Algorithm

Back Propagation (BP) scheme and the corresponding basic neural network training algorithm were proposed by Rumelhart, Hinton and Williams in 1986[65]. The training algorithm is done step by step. In each step, the BP scheme is first done layer by layer to

get the derivative information of cost function $E(\Phi)$ to weights Φ . The weights of the neural network Φ then are updated along the negative gradient direction in the weight space. The update formula is given by [65]

$$\Delta\Phi_{now} = \Phi_{next} - \Phi_{now} = -\eta \left. \frac{\partial E(\Phi)}{\partial \Phi} \right|_{\Phi=\Phi_{now}} \quad (2.20a)$$

or

$$\Delta\Phi_{now} = \Phi_{next} - \Phi_{now} = -\eta \left. \frac{\partial e_i(\Phi)}{\partial \Phi} \right|_{\Phi=\Phi_{now}} \quad (2.20b)$$

where constant η called learning rate controls the step size of weight update. In formula (2.20a), the weights are updated after all the training samples have been used to teach the neural network. This mode is called batch mode. Another mode represented by formula (2.20b) is called sample-by-sample mode where the weights are updated after each sample is presented to the network.

It is hard to determine the learning rate η in the basic BP algorithm described above. The smaller η is, the more iterations are needed for training. But if η is too large in order to speed up the learning process, the training may be unstable due to weight oscillation [84]. A simple method to improve this situation is the addition of a momentum term to weight update formulae as proposed by [65]

$$\Delta\Phi_{now} = -\eta \left. \frac{\partial E(\Phi)}{\partial \Phi} \right|_{\Phi=\Phi_{now}} + \alpha \Delta\Phi_{old} = -\eta \left. \frac{\partial E(\Phi)}{\partial \Phi} \right|_{\Phi=\Phi_{now}} + \alpha(\Phi_{now} - \Phi_{old}) \quad (2.21a)$$

$$\Delta\Phi_{now} = -\eta \left. \frac{\partial e_p(\Phi)}{\partial \Phi} \right|_{\Phi=\Phi_{now}} + \alpha \Delta\Phi_{old} = -\eta \left. \frac{\partial e_p(\Phi)}{\partial \Phi} \right|_{\Phi=\Phi_{now}} + \alpha(\Phi_{now} - \Phi_{old}) \quad (2.21b)$$

where constant α is the momentum factor which controls the influence of the last weight update direction on the current weight update, and Φ_{old} represents the last point of Φ . In

[84], it is pointed out that the momentum term has not been sufficiently utilized for reducing oscillation when there is a ravine in the error surface. A correction term that uses the difference of gradients is proposed to reduce weight oscillation. Another idea of reducing the weight oscillation around ravine of error surface was introduced by [85]. An extra constraint is placed so that the alignments of successive weight updates are maximized. The training becomes a constrained optimization problem and nonlinear programming techniques are used as the solution.

An interesting way to accelerate the convergence of BP is to use adaptation schemes that allow the learning rate and the momentum factor to be adaptive during training according to the training error [86]. An adaptive algorithm based on several heuristics is proposed by [87]. First, every weight in the neural network should have its own learning rate that is adaptive to the training process individually. Secondly, if a weight has kept changing in the same direction for several consecutive iterations, a larger step size needs to be applied. Otherwise if a weight is in an oscillation state for several iterations, a smaller step size should be used. In [88], an Enhanced Back Propagation algorithm is introduced. In this algorithm, the learning rate of a weight in the neural network will be increased if the weight is large and is going to be larger.

Each iteration of the training procedure can be considered as updating the weights in the neural network along a computed direction with a certain step size. The standard BP algorithm uses learning rate as the step size. A more efficient way is to choose an optimal step size using linear search methods along the computed update direction. Examples in this category are line search based on quadratic model [89], and line search based on linear interpolation [90].

2.3.3 Gradient-Based Optimization Methods

The standard BP and modified BP algorithms described in the above section are simple and relatively easy to implement. But there is no sound theory on choosing the values of training rate and momentum factor. The contradiction between slow convergence and oscillation around ravine area exists for most of the applications. Heuristics [86-88] are helpful to improve the situation. But implementing them in algorithms can be very difficult because there is no theory that can be used to determine the adaptation scheme. The algorithms using optimal step sizes [89,90] have a theoretical way to find a good step size. But the update direction simply derived from BP is not efficient. The training is as slow as standard BP algorithm when ravine exists in the error function. The best way would be determining the update direction and step size optimally simultaneously.

Because supervised training of neural networks can be considered as a functional optimization procedure, nonlinear high-order optimization methods using gradient information can be used to improve the rate of convergence. Compared to the algorithms based on heuristics, these methods have a sound theoretical basis and guaranteed convergence for most smooth functions. Some of the early work in this area was presented in [91], by developing second-order-training algorithms for neural networks. In [83], various first- and second-order optimization methods for feedforward neural networks training are reviewed.

As mentioned before, the cost function $E(\Phi)$ of training is defined as the summation of the square of difference between output of neural networks and training data. It is a

parametric function depending on all the training parameter in the neural network. The purpose of training is to find a (possibly local) minimum point $\Phi = \Phi^*$ in the parameter space that minimizes $E(\Phi)$. Because of the nonlinear activation function and the massive hidden neurons in the hidden layer, $E(\Phi)$ is generally complicated and nonlinear w.r.t. parameter Φ . Iterative descent algorithms are usually used in the training process. A general form of updating Φ can be described by (2.19). For the next point Φ_{next} , the following inequality should be satisfied

$$E(\Phi_{next}) = E(\Phi_{now} + \eta h) < E(\Phi_{now}) \quad (2.22)$$

Different optimization algorithms will have different ways to determine the direction vector h . The step size η is the optimal step size found by line search along h

$$\eta^* = \min_{\eta > 0} E(\Phi_{now} + \eta h) \quad (2.23)$$

If the direction vector h is determined based on the gradient (g) of the cost function $E(\Phi)$, the method is called a gradient-based descent method. In the following part, we describe two commonly used gradient-based methods, e.g., Conjugate Gradient method and quasi-Newton method.

A. Conjugate Gradient Method

Conjugate gradient method is originally derived from quadratic minimization. The direction vector h is chosen as the conjugate direction and is computed in a recursive way [92]. With an initial guess of weights $\Phi_{initial}$, the Conjugate Gradient method sets initial

gradient $\mathbf{g}_{initial} = \frac{\partial E}{\partial \Phi} \Big|_{\Phi=\Phi_{initial}}$, and direction vector $\mathbf{h}_{initial} = -\mathbf{g}_{initial}$. Then vector

sequences $\mathbf{g}$ and $\mathbf{h}$ are constructed recursively by [92]

$$\mathbf{g}_{next} = \mathbf{g}_{now} + \lambda_{now} \mathbf{H} \mathbf{h}_{now} \quad (2.24)$$

$$\mathbf{h}_{next} = -\mathbf{g}_{next} + \gamma_{now} \mathbf{h}_{now} \quad (2.25)$$

$$\lambda_{now} = \frac{\mathbf{g}_{now}^T \mathbf{g}_{now}}{\mathbf{h}_{now}^T \mathbf{H} \mathbf{h}_{now}} \quad (2.26)$$

$$\gamma_{now} = \frac{\mathbf{g}_{next}^T \mathbf{g}_{next}}{\mathbf{g}_{now}^T \mathbf{g}_{now}} \quad (2.27)$$

or,

$$\gamma_{now} = \frac{(\mathbf{g}_{next} - \mathbf{g}_{now})^T \mathbf{g}_{next}}{\mathbf{g}_{now}^T \mathbf{g}_{now}} \quad (2.28)$$

where $\mathbf{H}$ is the Hessian matrix of the objective function $E(\Phi)$. Equation (2.27) is called the Fletcher-Reeves formula while (2.28) is called the Polak-Ribiere formula [92].

To avoid the need of Hessian matrix for deriving the conjugate direction, an alternative way can be used to compute the conjugate direction. First compute Φ_{next} by proceeding Φ from Φ_{now} along the direction $\mathbf{h}_{now}$ to the local minimum through line

minimization, and then set $\mathbf{g}_{next} = \frac{\partial E}{\partial \Phi} \Big|_{\Phi=\Phi_{next}}$. Afterwards, This $\mathbf{g}_{next}$ is used as the result

vector of (2.24). In this way, only line minimization computation is needed to find the conjugate direction. As such, conjugate gradient methods become very efficient and scale well with the size of parameters in the neural network.

In standard conjugate gradient method, exact one-dimensional optimization is computational expensive because every function evaluation involves a complete cycle of sample presentation and neural network feed-forward calculation. To allow efficient

computation, inexact line search and modified conjugate direction were used. A Scaled Conjugate Gradient (SCG) algorithm was introduced in [93] to further reduce computational complexities by using Levenberg-Marquardt approach to scale the step size.

B. Quasi-Newton Method

Quasi-Newton method is also derived from quadratic objective function optimization. It has faster convergence rate than conjugate gradient method due to the proper estimation of inverse Hessian matrix. In this method, the descent direction $\mathbf{h}$ is computed by deflecting the gradient vector $\mathbf{g}$ using a matrix $\mathbf{C}$. Unlike Newton's method, the matrix $\mathbf{C}$ used here is an estimation of the inverse of Hessian matrix $\mathbf{H}^{-1}$. The weights are updated as [94]

$$\Phi_{next} - \Phi_{now} = -\eta \mathbf{h} = -\eta \mathbf{C}_{now} \mathbf{g}_{now} \quad (2.29)$$

$$\mathbf{C}_{now} = \mathbf{C}_{old} + \Delta \mathbf{C}_{now} \quad (2.30)$$

The $\mathbf{C}$ matrix is successively estimated employing rank 1 or rank 2 updates after each iteration [94]. To compute the term $\Delta \mathbf{C}_{now}$ in (2.30), there are mainly two rank 2 formulae,

$$\Delta \mathbf{C}_{now} = \frac{\Delta \Phi \Delta \Phi^T}{\Delta \Phi^T \Delta \mathbf{g}} - \frac{\mathbf{C}_{old} \Delta \mathbf{g} \Delta \mathbf{g}^T \mathbf{C}_{old}}{\Delta \mathbf{g}^T \mathbf{C}_{old} \Delta \mathbf{g}} \quad (2.31)$$

or

$$\Delta \mathbf{C}_{now} = \left(1 + \frac{\Delta \mathbf{g}^T \mathbf{C}_{old} \Delta \mathbf{g}}{\Delta \Phi^T \Delta \mathbf{g}}\right) \frac{\Delta \Phi \Delta \Phi^T}{\Delta \Phi^T \Delta \mathbf{g}} - \frac{\Delta \Phi \Delta \mathbf{g}^T \mathbf{C}_{old} + \mathbf{C}_{old} \Delta \mathbf{g} \Delta \Phi^T}{\Delta \Phi^T \Delta \mathbf{g}} \quad (2.32)$$

where

$$\Delta\Phi = \Phi_{now} - \Phi_{old}, \quad \Delta\mathbf{g} = \mathbf{g}_{now} - \mathbf{g}_{old} \quad (2.33)$$

Equation (2.31) is called the DFP (Davidon-Fletcher-Powell) formula and equation (2.32) is called the BFGS (Broyden-Fletcher-Goldfarb-Shanno) formula [94].

Suppose there are total N_ϕ weights in the neural network structure. For standard quasi-Newton methods, N_ϕ^2 storage space and a line search are required to maintain an approximation of the inverse Hessian matrix and calculate a reasonably accurate step size. In Limited-memory (LM) or one-step BFGS method [95], the inverse Hessian approximation is reset to the identity matrix $\mathbf{I}$ after every iteration. The space of storing matrices is consequently saved. As N_ϕ increases, the consumption of storage memory and computation on line search become the bottleneck for efficiency and scaling. In [96], a second-order learning algorithm is proposed so that the descent direction $\mathbf{h}$ is computed on the basis of a partial BFGS update with less memory. A reasonably accurate step size is efficiently calculated as the minimal point of a second-order approximation of the cost function. In [97], various approaches to the parallel implementation of second-order gradient-based MLP training algorithms, such as full and limited memory BFGS algorithms, were presented.

2.3.4 Global Optimization Methods

All gradient-based training methods share the defect of converging to a local minimum as the training result. To allow the training algorithms to escape from local minima and converge to the global minimum, some training methods using random optimization techniques have been proposed. These methods are characterized by a

random search element in the training process. Two representative algorithms in this class are the simulated annealing algorithm and the genetic algorithm. In [66], the simulated annealing algorithm is described in such a manner that analogs the annealing process of atoms inside metal. The random parameter in the optimization process is controlled by a parameter called temperature, which determines the search range a local minimum can jump out. In [67], the genetic training algorithm is introduced to train the neural network. The genetic algorithm imitates the biological evolution and the weights of the neural network are coded as chromosomes. The local convergence of the algorithm is achieved by crossover operation and selection procedure, and the global search is accomplished by the random mutation. In [98], based on a Langevin updating (LV) rule, noise is added to the weights during training. And in [99], a stochastic minimization algorithm is developed for neural networks training.

Because of the random search during the neural network training, the convergence of these training methods is quite slow. A hybrid method that combines the conjugate gradient method and random optimization is proposed in [89]. During the training with conjugate gradient method, if a flat error surface is encountered, the training algorithm switches to the random optimization method. After the training escapes from the flat error surface, it switches back to the conjugate gradient algorithm.

2.4 Review of Adjoint Sensitivity Techniques

Adjoint technique is one of the most efficient techniques for the sensitivity analysis. In this section, the fundamental concepts of adjoint networks are first reviewed.

Following that, several important applications of adjoint networks and sensitivities are presented.

2.4.1 The Fundamentals of Adjoint Network and Sensitivities

The first adjoint network approach for gradient computation was completely freed from a special analysis scheme by Director and Rohrer in 1969 [100]. All the sensitivity expressions could be evaluated after only two network analyses: one from the original network and one from the adjoint network. Here, a brief summary of the results from this classic approach is given.

Consider a general network N and let the branch voltages and currents of N be represented by $v_b(t)$ and $i_b(t)$ respectively. Let $\tilde{N}$ denote the adjoint network related to the original network N , $\psi_b(\tau)$ and $\phi_b(\tau)$ denote the branch voltages and currents in the adjoint network. Assume a perturbation in the element values of network N , and let $\Delta v_b(t)$ and $\Delta i_b(t)$ be such variations in the branch voltages and currents. As Tellegen's theorem [101] is independent of element values, one can observe that

$$\sum_b [\Delta v_b(t) \phi_b(\tau) - \Delta i_b(t) \psi_b(\tau)] \equiv 0 \quad (2.34)$$

Based on (2.34), the adjoint network branch descriptions and the general adjoint sensitivities for various circuit elements can be subsequently derived and results are summarized in Table 2-I [100]. In those formulae, $x_r(t)$, $x_c(t)$ and $x_L(t)$ are the chosen representation parameters. p_r , p_c and p_L represent typical design parameters. In

addition, $q_c(t)$ is the capacitance charge and $\lambda_L(t)$ is the inductance flux. Such subscripts R , C and L denote resistance element, capacitance element and inductance element respectively.

During the automated design of networks containing nonlinear elements, multidimensional optimization technique has to be used to find the suitable values of designable parameters to minimize the performance function [102]. In a nonlinear design problem, a time-domain performance function could be formed [100][103] as

$$\begin{aligned} \varepsilon = & \sum_V \int_{t_0}^{t_f} \frac{1}{2} w_V(t) \left[(i_V(t) - \hat{i}_V(t)) \right]^p dt \\ & + \sum_I \int_{t_0}^{t_f} \frac{1}{2} w_I(t) \left[(v_I(t) - \hat{v}_I(t)) \right]^p dt \end{aligned} \quad (2.35)$$

where the $\hat{i}_V(t)$ is the desired port voltage source current responses, $i_V(t)$ the actual port voltage source current responses, $\hat{v}_I(t)$ the desired port current source voltage responses, $v_I(t)$ the actual port current source voltage responses. The $w_V(t)$ and $w_I(t)$ are the nonnegative weighting functions, which give more flexibility for the design.

By supplying a suitable set of adjoint network excitations, we can obtain the following gradients components [100]:

$$\begin{aligned} \frac{\partial \varepsilon}{\partial p_R} = & \int_{t_0}^{t_f} \left[\frac{\partial g_R(x_R(t), p_R, t)}{\partial p_R} \psi_R(t_0 + t_f - t) \right. \\ & \left. - \frac{\partial f_R(x_R(t), p_R, t)}{\partial p_R} \phi_R(t_0 + t_f - t) \right] dt \end{aligned} \quad (2.36)$$

for resistive components;

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial p_c} = \int_{t_0}^{t_f} & \left[\frac{d}{dt} \frac{\partial g_c(x_c(t), p_c, t)}{\partial p_c} \psi_c(t_0 + t_f - t) \right. \\ & \left. - \frac{\partial f_c(x_c(t), p_c, t)}{\partial p_c} \phi_c(t_0 + t_f - t) \right] dt \end{aligned} \quad (2.37)$$

for capacitive components;

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial p_L} = \int_{t_0}^{t_f} & \left[\frac{\partial g_L(x_L(t), p_L, t)}{\partial p_L} \psi_L(t_0 + t_f - t) \right. \\ & \left. - \frac{d}{dt} \frac{\partial f_L(x_L(t), p_L, t)}{\partial p_L} \phi_L(t_0 + t_f - t) \right] dt \end{aligned} \quad (2.38)$$

for inductive components. With these gradient elements, a number of optimal design algorithms can follow immediately.

This original work was focused on time-domain case. In [104], the adjoint sensitivities for frequency domain automated network design were presented. Another extension of this approach for use in the sensitivity calculations for noncommensurate networks was given in [105]. For a class of nonlinear networks, the explicit expressions for sensitivity of a response due to nonlinear elements were derived in [106]. Subsequently, utilizing these expressions, certain invariance relations were also established.

Table 2-I: The summaries of the adjoint network branch relations and sensitivity coefficients

Element Type	Parametric Representation	Adjoint Description	Sensitivity
Resistive	$v_R(t) = f_R(x_R(t), p_R, t)$ $i_R(t) = g_R(x_R(t), p_R, t)$	$\frac{\partial f_R}{\partial x_R} \phi_R(\tau) = \frac{\partial g_R}{\partial x_R} \psi_R(\tau)$	$-\frac{\partial f_R}{\partial p_R} \phi_R(\tau) + \frac{\partial g_R}{\partial p_R} \psi_R(\tau)$
Capacitive	$v_c(t) = f_c(x_c(t), p_c, t)$ $i_c(t) = g_c(x_c(t), p_c, t)$ $i_c(t) = \frac{d}{dt} q_c(t)$	$\frac{\partial f_c}{\partial x_c} \phi_c(\tau) = \frac{\partial g_c}{\partial x_c} \frac{d}{d\tau} \psi_R(\tau)$	$-\frac{d}{dt} \left[\frac{\partial g_c}{\partial p_c} \right] \psi_c(\tau) - \frac{\partial f_c}{\partial p_c} \phi_c(\tau)$
Inductive	$v_L(t) = \frac{d}{dt} \lambda_L(t)$ $\lambda_L(t) = f_L(x_L(t), p_L, t)$ $i_L(t) = g_L(x_L(t), p_L, t)$	$\frac{\partial f_L}{\partial x_L} \frac{d}{d\tau} \phi_L(\tau) = \frac{\partial g_L}{\partial x_L} \psi_L(\tau)$	$\frac{\partial g_L}{\partial p_L} \psi_L(\tau) - \frac{d}{dt} \left[\frac{\partial f_L}{\partial p_L} \right] \phi_L(\tau)$

2.4.2 Important Applications of Adjoint Techniques

A. HB-Based Adjoint Technique

The HB method has become an important tool for large-signal steady-state periodic analysis of nonlinear circuits. The HB analysis is carried out in frequency domain and the signal responses are generated from their spectrums. In the steady-state case, the time domain approach, such as that of SPICE, however, is very cumbersome since it has to be performed until the transient effects vanish. Therefore, the existing adjoint nonlinear transient sensitivity approach [107], if still applicable, would be very inefficient as well. On the other hand, nonlinear DC analysis cannot be performed separately from the ac analysis if large signal ac sources exist in the circuit. This fact indicates that such nonlinear dc adjoint technique [107] is not directly applicable to the HB technique. The work of Bandler *et al.* [108] firstly expended the HB technique with the powerful adjoint network concept. It provided an efficient tool for the otherwise computationally expensive but essential gradient computation in the nonlinear design and optimization involving HB method. It has been reported in [108] this sensitivity analysis approach can lead to a tremendous CPU time saving, i.e. 90 percent for both simulation and exact sensitivity computations and 98 percent for sensitivity analysis only, when applied to a *MESFET* mixer example. This unified theory can be immediately utilized in the parameter extraction of nonlinear devices under RF large-signal excitations by providing fast and accurate gradient information.

B. EM-Based Adjoint Technique

As satellite communication antennas require keen performance in terms of return loss, polarization purity and high power handling capability, all the subsystems, including the feed, have to be designed through the efficient full-wave electromagnetic simulators. The work of Mongiardo *et al.* [109] firstly introduced the adjoint network method into the automated design of corrugated feeds. Generally full-wave design of corrugated feeds is accomplished by repeated numerical analysis of different feed geometries, which are obtained by slightly varying the corrugation geometrical sizes. Since hundreds of discontinuities exist in a corrugated feed, the above procedure is computationally very expensive. In order to reduce the numerical efforts, the adjoint network method together with the generalized scattering matrix approach has been applied to the design and sensitivity analysis of circular corrugated feeds. With this combined approach, the return loss sensitivities with respect to all geometrical sizes are computed with just one analysis of the entire feed. The efficiency of this approach is illustrated by the design example of feeds with more than 100 corrugations.

Historically, the adjoint variable method (AVM) has attracted very little attention in full-wave EM design. The recent work of [110] focused on the adjoint-based sensitivity analysis for gradient optimization with full-wave frequency-domain EM solver. It demonstrated that through this feasible technique the construction of the adjoint system required only minor modifications of the existing codes of method of moments. After original problem is solved, the solution to the adjoint problem is obtained with just little overhead. Consequently, the gradient of

objective function is computed through only a single analysis regardless of number of design parameter. In this way, the AVM approach substantially increases the efficiency of frequency domain full-wave CAD tools.

C. Static ANN-Based Adjoint Technique

Over the decades, adjoint network approach has gained attention as an important vehicle for efficient sensitivity analysis in the circuit community [79,107]. It has become a mature technique for the sensitivity evaluation of a variety of linear and nonlinear circuits in frequency domain or in time domain. As ANNs have been introduced for RF and microwave modeling and design, neural based sensitivity analysis has become an essential step to perform neuro-based circuit design. In [111], an adjoint neural network sensitivity analysis technique is presented to facilitate exact sensitivity calculations in a generic feed forward neural network structure accommodating empirical functions, equivalent circuits and the traditional switch type neurons.

Two models, one called the original neural network model, and the other defined as the adjoint neural network model, are utilized in this sensitivity analysis technique. The original neural network represents a generic framework in which microwave knowledge functions [7] can be coherently embedded in the neural network structures, and connections between neurons can be arbitrary allowing different types of microwave neural structures to be included. Two types of adjoint neural networks are formulated. The basic adjoint neural network can be conveniently constructed by flipping the original neural model between input and output [111]. To create a trainable adjoint neural network structure, a set of fictitious neurons called Element Derivative Neuron (EDN)

[111] are introduced as the bridge between the original neural model and adjoint neural model. The EDNs are stimulated by the neurons of the original neural network and their responses become the stimulus to the adjoint model.

Let x and y represent inputs and outputs of the original neural network model. An algorithm for combined training of the adjoint and the original neural models [111] is formulated to let the neural model fit not only the x - y relationship, but simultaneously also the required derivative relationship of y w.r.t x . A unified training error is defined with the consideration of both original training error and adjoint training error. During training, both the original and the adjoint neural models share the same set of weight parameters, therefore training one model will also result in the other model being updated. This method is very useful in simultaneously DC/small-signal/large signal device modeling [111] and neural-based microwave circuit design.

2.5 Conclusions

Recently, neural networks have been extensively used in the modeling, simulation and optimization of RF/Microwave components and circuits. In this chapter, we have presented various kinds of neural network structures, which have been widely adopted in microwave ANN area. As training is a key step in neural model development, different neural network training algorithms, including basic BP scheme and sophisticated gradient-based methods as well as non-gradient methods have been described in depth. Furthermore, existing adjoint techniques in such areas as circuit, EM and ANN have been reviewed to illustrate the efficiency and flexibility of adjoint concept. However, none of existing neural network approaches can be applied to model nonlinear transient circuit

behaviors such as those in high-speed interconnect with nonlinear buffers. In the following part of the thesis, a new adjoint sensitivity technique is proposed to combine the elegant adjoint concept with merits of ANN-based technique yielding neuro-based transient modeling and design.

Chapter 3

Proposed Adjoint Dynamic Neural Network Technique

3.1 Introduction

In the last chapter, we reviewed the state-of-the-art of neural network techniques in RF/Microwave modeling and design. A most recent trend in this area is the use of a unique category of ANN called dynamic neural networks (DNN) [4][112] for dynamic nonlinear CAD. In [4], DNNs have been shown to address large-signal modeling and design in the case of nonlinear periodic RF/microwave responses in harmonic balance (HB) environment [108][113]. Unlike the feedforward neural networks such as the popularly used multiplayer perceptron (MLP) [1][2] which models nonlinear algebraic relationships, the DNN addresses nonlinear dynamic relationships in circuits and systems. Compared to the recurrent neural networks [3][78], which are often used in signal processing [78][114] and control [115], the DNN approach is more compatible with nonlinear circuit simulation requirements and more effective in addressing the challenges in nonlinear circuit modeling. This recent development has led to DNNs being applied to nonlinear behavioral modeling of amplifiers [4, 116] and mixers [3][4] and the trained

DNN behavioral models are incorporated into system simulator [80] to perform harmonic balance and envelop analysis of DBS receiver subsystems [4].

In this chapter, we further expand DNN into an even more powerful technique that can handle nonlinear transients. Accurate and fast representation of nonlinear transient behaviors is a key to successful digital high-speed VLSI interconnect CAD including multi-chip modules and multi-layer printed circuit boards (PCB) [117]. Extensive research has been conducted on modeling and simulation of interconnect networks resulting in circuit-based [118], model-reduction-based [119] and ANN-based techniques [20]. Sensitivity analysis of transmission lines [120,121] and transient-oriented optimization [11,122,123] of VLSI interconnects have been used in the design of high-speed packages improving signal integrity. Currently, CAD of interconnect modules with nonlinear terminations is an active research subject [119-121,124] and this thesis is targeted towards accomplishing efficient neural based nonlinear transient modeling and design of high-speed interconnect components including physics-based effects.

In order for the DNN to learn transient data, sensitivities (derivatives) of the corresponding training error w.r.t. DNN weights are essential. In addition, transient design requires sensitivities of the target functions w.r.t. geometrical/physical parameters of nonlinear components. However, due to the dynamic nature of transient response and the DNN being not directly algebraic, the sensitivity information cannot be obtained by a simple differentiation of the DNN equations. A brute force approximation is the perturbation approach where the DNN differential equations have to be re-solved for the perturbation of each DNN variable. For DNN training with transient data and many internal weight

variables, this would lead to substantially long training time. This is one of the major factors making circuit transient modeling by neural networks still a much less explored subject.

Our motivation towards solving this bottleneck problem is the paramount sensitivity concept, namely, the adjoint sensitivity concept pioneered in [100]. Following this, various adjoint sensitivity techniques such as HB-based [108], circuit-based [79], transmission line-based [121], EM-based [109][110], and static ANN-based [111] techniques have been developed. In this thesis, exact adjoint sensitivity for DNN is investigated to enable transient modeling and design.

For the first time, we propose a new adjoint dynamic neural network (ADNN) technique for nonlinear transients such as those in high-speed interconnects with nonlinear terminations. An elegant formulation for exact adjoint sensitivities is derived employing Lagrange functions approach, by defining an adjoint of the DNN. These formulae are exploited to engineer a robust training algorithm that facilitates efficient DNN training directly from transient data. The technique is also generalized to cover DNN sensitivities w.r.t. external static inputs (such as physical/geometrical values of components) to the model leading to faster nonlinear circuit design including geometrical/physical parameters.

The rest of the chapter is organized as follows. Chapter 3.2 gives a brief description of the DNN formulation for modeling nonlinear dynamic circuits. In Chapter 3.3, the proposed ADNN is presented and the exact sensitivities of the energy function for DNN system with respect to both DNN internal weights and external inputs are systematically derived. As a direct application of the resulting sensitivity formulae, transient-oriented

DNN training algorithm is described in Chapter 3.4. Finally, in Chapter 3.5 we summarize the proposed ADNN technique.

3.2 Dynamic Neural Network

In time-domain, a nonlinear dynamic circuit can be represented by a DNN [4] of order n . Inputs to DNN include dynamic inputs $\mathbf{u}(t)$, corresponding k^{th} -order derivatives $\mathbf{u}^{(k)}(t)$, and static inputs $\mathbf{s}$. Here, $\mathbf{s}$ is a N_s vector containing static inputs such as geometrical/physical parameters of the circuit to be modeled. Let N_y be the number of DNN outputs. The DNN equations of the nonlinear model are given by,

$$\begin{aligned}\dot{\mathbf{v}}_1(t) &= \mathbf{v}_2(t) \\ &\vdots \\ \dot{\mathbf{v}}_{n-1}(t) &= \mathbf{v}_n(t) \\ \dot{\mathbf{v}}_n(t) &= \mathbf{f}_{\text{ANN}}(\mathbf{s}, \mathbf{u}^{(n-1)}(t), \dots, \mathbf{u}^{(1)}(t), \mathbf{u}(t), \mathbf{v}_n(t), \dots, \mathbf{v}_1(t) \mid \mathbf{w})\end{aligned}\tag{3.1}$$

where each $\mathbf{v}_i$ is a N_y -vector representing a set of states of the DNN. Output signals of the DNN model are given by $\mathbf{y}(t) = \mathbf{v}_1(t)$. Here, $\mathbf{f}_{\text{ANN}}$ represents a Multilayer Perceptron (MLP) neural network [1][2] with trainable weight parameters $\mathbf{w}$.

MLP neural network is a popularly used neural network structure for RF/Microwave circuit modeling and design. Let L be the total number of layers of an MLP neural network. Let the number of neurons in l^{th} layer be N_l , $l = 1, 2, \dots, L$. Let w_{ij}^l represent the weight of the link between j^{th} neuron of $l-1^{\text{th}}$ layer and i^{th} neuron of l^{th} layer. Let x_i represent the i^{th} external input to the MLP and z_i^l be the output of i^{th} neuron of l^{th} layer. In general, trainable parameters $\mathbf{w}$ of MLP includes

$w_{ij}^l, j=1, \dots, N_{l-1}, i=1, 2, \dots, N_l, l=2, 3, \dots, L$, i.e., $\mathbf{w} = [w_{11}^2 \ w_{12}^2 \ \dots \ w_{N_L N_{L-1}}^L]$. Let q

and r denote the number of inputs and outputs of MLP. In our DNN case, inputs to the MLP are defined as

$$\mathbf{x} = [x_1 \ x_2 \ \dots \ x_q]^T = [s, (\mathbf{u}^{(n-1)})^T, \dots, (\mathbf{u}^{(1)})^T, \mathbf{u}^T, \mathbf{v}_n^T, \dots, \mathbf{v}_1^T]^T. \quad (3.2)$$

Neural network feedforward computation is a process used to compute the output vector

$\mathbf{y} = [y_1 \ y_2 \ \dots \ y_r]^T$. For our f_{ANN} structure, the feedforward computation can be described by

$$z_i^l = x_i, \ i=1, 2, \dots, N_1, q=N_1, l=1 \quad (3.3)$$

$$z_i^l = \sigma\left(\sum_{j=1}^{N_{l-1}} w_{ij}^l z_j^{l-1}\right), \ i=1, 2, \dots, N_l, l=2, 3, \dots, L-1 \quad (3.4)$$

$$y_i = \sum_{j=1}^{N_{L-1}} w_{ij}^L z_j^{L-1}, \ i=1, 2, \dots, N_L, r=N_L, l=L \quad (3.5)$$

where $\sigma(\cdot)$ is a sigmoid switching function. During the feedforward process, the weight parameters $\mathbf{w}$ are fixed.

3.3 Proposed ADNN Sensitivity Technique

3.3.1 Problem Statement

We define an energy function E for DNN that represents a typical design function in transient analysis [79] as

$$E = \int_{t_1}^{t_2} f(y(t)) \ dt \quad (3.6)$$

where $[T_1, T_2]$ is the time-interval of interest. The purpose of our sensitivity technique is to determine the derivatives of E w.r.t. DNN internal weights $\mathbf{w}$ and static inputs $\mathbf{s}$, i.e., $\frac{dE}{d\mathbf{w}}$ and $\frac{dE}{d\mathbf{s}}$. Such derivatives could be very useful for training DNN to learn nonlinear transient data and for sensitivity analysis and optimization of high-speed interconnect circuits utilizing the trained DNN model. The challenge here is that, $\mathbf{y}(t)$ has a dynamic and not a simple algebraic relationship with $\mathbf{w}$ and $\mathbf{s}$. In this thesis, we use the adjoint concept to derive an efficient sensitivity technique for transient oriented DNN training and applications.

3.3.2 Proposed ADNN

To derive the DNN sensitivity, we utilize the concept of Lagrange function and formulate our task as a nonlinear optimization problem of (3.6) with DNN equations (3.1) as dynamic constraints. We define a Lagrange function L as

$$L = f(\mathbf{y}(t)) + \hat{\mathbf{v}}_n^T (\dot{\mathbf{v}}_n - \mathbf{f}_{\text{ANN}}) + \sum_{j=1}^{n-1} \hat{\mathbf{v}}_j^T (\dot{\mathbf{v}}_j - \mathbf{v}_{j+1}) \quad (3.7)$$

where each $\hat{\mathbf{v}}_j$ is a vector representing time-dependent Lagrange parameter, independent of $\mathbf{w}$ and $\mathbf{s}$. Subject to (3.1), derivatives of E can be expressed in terms of L in (3.7) as

$$\begin{aligned} \frac{dE}{d\mathbf{w}} = & \int_{T_1}^{T_2} \hat{\mathbf{v}}_n^T \frac{d\dot{\mathbf{v}}}{d\mathbf{w}} dt - \int_{T_1}^{T_2} \sum_{j=1}^{n-1} \hat{\mathbf{v}}_j^T \frac{d\mathbf{v}_{j+1}}{d\mathbf{w}} dt \\ & - \int_{T_1}^{T_2} \hat{\mathbf{v}}_n^T \left(\frac{\partial \mathbf{f}_{\text{ANN}}}{\partial \mathbf{v}^T} \frac{d\mathbf{v}}{d\mathbf{w}} + \frac{\partial \mathbf{f}_{\text{ANN}}}{\partial \mathbf{w}} \right) dt + \int_{T_1}^{T_2} \frac{\partial f}{\partial \mathbf{y}^T} \frac{d\mathbf{y}}{d\mathbf{w}} dt \end{aligned} \quad (3.8)$$

Integrating the first term in (3.8) by parts and noticing that $\mathbf{y}(t) = \mathbf{v}_1(t)$, we can obtain

$$\begin{aligned}
\frac{dE}{dw} = & \hat{\mathbf{v}}(t)^T \frac{d\mathbf{v}(t)}{dw} \Big|_{T_2}^{T_1} - \int_{T_1}^{T_2} \hat{\mathbf{v}}_n^T \frac{\partial f_{\text{ANN}}}{\partial w} dt + \int_{T_1}^{T_2} \left(\frac{\partial f}{\partial \mathbf{y}^T} - \hat{\mathbf{v}}_1^T - \hat{\mathbf{v}}_n^T \frac{\partial f_{\text{ANN}}}{\partial \mathbf{v}_1^T} \right) \frac{d\mathbf{v}_1}{dw} dt \\
& + \int_{T_1}^{T_2} \sum_{j=2}^n \left(-\hat{\mathbf{v}}_j^T - \hat{\mathbf{v}}_n^T \frac{\partial f_{\text{ANN}}}{\partial \mathbf{v}_j^T} - \hat{\mathbf{v}}_{j-1}^T \right) \frac{d\mathbf{v}_j}{dw} dt
\end{aligned} \tag{3.9}$$

As (3.9) holds good for arbitrary choice of $\hat{\mathbf{v}}_j$'s, we propose a new Adjoint-DNN or ADNN as

$$\begin{aligned}
\dot{\hat{\mathbf{v}}}_1 &= -\frac{\partial f_{\text{ANN}}^T}{\partial \mathbf{v}_1} \hat{\mathbf{v}}_n + \frac{\partial f}{\partial \mathbf{y}} \\
\dot{\hat{\mathbf{v}}}_2 &= -\frac{\partial f_{\text{ANN}}^T}{\partial \mathbf{v}_2} \hat{\mathbf{v}}_n - \hat{\mathbf{v}}_1 \\
&\vdots \\
\dot{\hat{\mathbf{v}}}_n &= -\frac{\partial f_{\text{ANN}}^T}{\partial \mathbf{v}_n} \hat{\mathbf{v}}_n - \hat{\mathbf{v}}_{n-1}
\end{aligned} \tag{3.10}$$

where each $\hat{\mathbf{v}}_j$ is a N_y -vector representing a set of ADNN states. The boundary conditions for equations (3.10) are imposed at the upper time limit as

$$\hat{\mathbf{v}}(T_2) = \mathbf{0}. \tag{3.11}$$

As can be observed from (3.1) and (3.10), both DNN and corresponding ADNN have the same number of states. Input to ADNN (i.e. $\frac{\partial f}{\partial \mathbf{y}}$) excites state $\hat{\mathbf{v}}_1$ that corresponds to output of the original-DNN. Outputs of the ADNN are given by $\hat{\mathbf{y}}(t) = \hat{\mathbf{v}}_n(t)$. ADNN is a linear time-varying system, which is simulated backward in time from T_2 to T_1 , in contrast to the original nonlinear DNN, which is solved forward in time from T_1 to T_2 .

3.3.3 Calculations of Derivatives

Using ADNN of (3.10) together with the boundary condition of (3.11), Equation (3.9) can be re-written as

$$\frac{dE}{dw} = \hat{v}^T(T_1) \frac{dv(T_1)}{dw} - \int_{T_1}^{T_2} \hat{v}_n^T \frac{\partial f_{ANN}}{\partial w} dt \quad (3.12)$$

In general, the initial conditions of original DNN (i.e., $v(T_1)$) are independent of w and s .

Now $\frac{dE}{dw}$ can be expressed in terms of ADNN output $\hat{y}(t)$ and MLP network f_{ANN} in original DNN as

$$\frac{dE}{dw} = - \int_{T_1}^{T_2} \hat{y}^T \frac{\partial f_{ANN}}{\partial w} dt \quad (3.13)$$

where $\frac{\partial f_{ANN}}{\partial w}$ is a Jacobian matrix containing derivatives of the MLP w.r.t individual weight parameters in w .

The expression (3.13) could be further specialized according to the locations of weight parameters in the MLP part of DNN. For example, consider f_{ANN} being a 3-layer MLP. Let N_l denote number of neurons in l^{th} layer. Let w_{ij}^l represent weight of the link between j^{th} neuron of $l-1^{\text{th}}$ layer and i^{th} neuron of l^{th} layer. We define $z_i^l(t)$ as instantaneous output of the i^{th} neuron in the l^{th} layer. Sensitivities in (3.13) can then be systematically evaluated as

$$\frac{dE}{dw_{ij}^l} = \begin{cases} - \int_{T_1}^{T_2} \hat{y}_k(t) z_i^{l-1}(t) dt, & l = 3, k = i, k = 1, 2, \dots, N_y \\ - \sum_{k=1}^{N_y} w_{ki}^{l+1} \int_{T_1}^{T_2} \hat{y}_k(t) z_i^l(t) (1 - z_i^l(t)) z_j^{l-1}(t) dt, & l = 2 \end{cases} \quad (3.14)$$

where $\hat{y}_k(t)$ represents k^{th} ADNN output. To derive the expression for $\frac{dE}{ds}$, we choose the location of static DNN inputs $\mathbf{s}$ within MLP input vector $\mathbf{x}$ as $s_i = x_i$, $i = 1, 2, \dots, N_s$. Following a similar derivation, sensitivity of E w.r.t. i^{th} static DNN input s_i can be evaluated using

$$\frac{dE}{ds_i} = - \sum_{k=1}^{N_y} \sum_{j=1}^{N_l} w_{kj}^{l+1} w_{ji}^l \int_{T_1}^{T_2} \hat{y}_k(t) z_j^2(t) (1 - z_j^2(t)) dt, \quad l = 2. \quad (3.15)$$

3.4 ADNN Sensitivities for DNN Training

3.4.1 Training Objective

Our elegant choice of E allows direct utilization of ADNN sensitivity formulation in Chapter 3.3, for DNN training using nonlinear transients. Let $\mathbf{u}_d^i(t)$ and $\mathbf{y}_d^i(t)$ represent i^{th} input and output waveforms sufficiently sampled in the time-interval $[T_1, T_2]$, to be used as training data. Then for i^{th} training waveform pair $(\mathbf{u}_d^i(t), \mathbf{y}_d^i(t))$, per waveform training error can be defined as

$$E_{d_i} = \frac{1}{2} \int_{T_1}^{T_2} \left\| \mathbf{y}^i(t) - \mathbf{y}_d^i(t) \right\|^2 dt \quad (3.16)$$

The objective of DNN training is to adjust DNN parameters $\mathbf{w}$ such that the total training error function

$$E_d = \sum_{i=1}^{N_r} E_{d_i} = \int_{T_1}^{T_2} \sum_{i=1}^{N_r} \frac{1}{2} \left\| \mathbf{y}^i(t) - \mathbf{y}_d^i(t) \right\|^2 dt \quad (3.17)$$

is minimized. Here, $y^i(t)$ represents DNN prediction of i^{th} output signal and N_T represents total number of training waveforms pairs $(u_d^i(t), y_d^i(t))$. Since f in (3.6) can be any function, we judiciously choose $f(y(t))$ as

$$f(y(t)) = \sum_{i=1}^{N_T} \frac{1}{2} \|y^i(t) - y_d^i(t)\|^2 \quad (3.18)$$

to establish consistency between energy function E and training error E_d . Dynamic training process for i^{th} training waveform pair $(u_d^i(t), y_d^i(t))$ is briefly explained here. Given $u_d^i(t)$, original DNN equations in (3.1) are integrated forward from time T_1 up to T_2 with user-specified initial condition $v(T_1)$, resulting in current DNN outputs $y^i(t)$. To evaluate ADNN equations, the adjoint excitation can be formulated in terms of DNN outputs $y^i(t)$ as

$$\frac{\partial f}{\partial y} = y^i(t) - y_d^i(t). \quad (3.19)$$

We substitute (3.19) into (3.10) and then integrate ADNN equations in (3.10) backwards from time T_2 to T_1 , setting $\hat{v}(T_2) = \mathbf{0}$. Finally, dynamic error sensitivities computed using (3.13) are supplied to a gradient-based training algorithm (e.g. quasi-Newton) for determining the weight update during DNN training process shown in Figure 3.1.

Figure 3.1: Block diagram of the ADNN enabled training algorithm used to train DNN to learn transient waveform data. D is an operator for time-differentiation. Both the original DNN and the proposed ADNN are evaluated using numerical integration in each epoch of training, providing the information of training error and its derivatives w.r.t. DNN internal weights w .

3.4.2 Further Theoretical Relations between ADNN and DNN

We examine how the ADNN is solved and relate it to the solution process of the original DNN. Since we are dealing with transient responses, we need to use numerical techniques for solving the differential equations to evaluate the original and adjoint dynamic neural models. Here we employ the most frequently used type of such integration techniques, i.e., implicit integration methods [79]. Applying a general multistep implicit integration formula [107], DNN equations (3.1) can be reduced to a set of nonlinear algebraic equations given by

$$\begin{bmatrix} v_1(t_{k+1}) \\ \vdots \\ v_{n-1}(t_{k+1}) \\ v_n(t_{k+1}) \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^m [a_i v_1(t_{k-i}) + \Delta t b_i v_2(t_{k-i})] \\ \vdots \\ \sum_{i=0}^m [a_i v_{n-1}(t_{k-i}) + \Delta t b_i v_n(t_{k-i})] \\ \sum_{i=0}^m [a_i v_n(t_{k-i}) + \Delta t b_i f_{\text{ANN}}(t_{k-i})] \end{bmatrix} + \Delta t b_{-1} \begin{bmatrix} v_2(t_{k+1}) \\ \vdots \\ v_n(t_{k+1}) \\ f_{\text{ANN}}(t_{k+1}) \end{bmatrix}$$

$$f_{\text{ANN}}(t_j) = f_{\text{ANN}}(s, u^{(n-1)}(t_j), \dots, u^{(1)}(t_j), u(t_j), v_n(t_j), \dots, v_1(t_j) \mid w) \quad (3.20)$$

for $j = k - m, k - m + 1, \dots, k + 1$

where t_k and t_{k+1} represent two consecutive time instances and $\Delta t = t_{k+1} - t_k$ is the time step for forward integration. Here, m represents the order of the integration formula while a_i and b_i are real coefficients defining the specific implicit integration method. To solve such nonlinear equations, Newton-Raphson (NR) iterative techniques are commonly used [107]. For each iteration, the solution process involves the LU factorizing of the Jacobian matrix representing the derivatives of the equations in (3.20) w.r.t. the DNN state variables at time t_{k+1} , i.e.,

$$J = \begin{bmatrix} I & -\Delta t b_{-1} I & 0 & \dots & 0 \\ 0 & I & -\Delta t b_{-1} I & \dots & 0 \\ & & \vdots & & \\ 0 & \dots & 0 & I & -\Delta t b_{-1} I \\ -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_1}(t_{k+1}) & -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_2}(t_{k+1}) & \dots & -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_{n-1}}(t_{k+1}) & -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_n}(t_{k+1}) + I \end{bmatrix} \quad (3.21)$$

where I denotes a $N_y \times N_y$ identity matrix. Now we examine the solution process for the ADNN, which can be solved using implicit integration backward in time from T_2 to T_1 . Applying the multistep implicit integration formula to ADNN of (3.10), we obtain time-varying linear algebraic equations

$$\begin{bmatrix} I & 0 & 0 & -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_1}(t_{k+1}) \\ -\Delta t b_{-1} I & I & \vdots & -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_2}(t_{k+1}) \\ 0 & -\Delta t b_{-1} I & \dots & 0 \\ \vdots & \vdots & I & -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_{n-1}}(t_{k+1}) \\ 0 & 0 & -\Delta t b_{-1} I & -\Delta t b_{-1} \frac{\partial f_{\text{ANN}}^T}{\partial v_n}(t_{k+1}) + I \end{bmatrix} \begin{bmatrix} \hat{v}_1(t_{k+1}) \\ \hat{v}_2(t_{k+1}) \\ \vdots \\ \hat{v}_{n-1}(t_{k+1}) \\ \hat{v}_n(t_{k+1}) \end{bmatrix} = c \quad (3.22)$$

where c is a nN_y -vector determined by the ADNN solutions at previous time instances,

i.e., $t = t_{k-m}, \dots, t_k$ and present ADNN input $\frac{\partial f}{\partial y}(t_{k+1})$. In this way, all the state variables

$\hat{v}_1(t_{k+1}), \hat{v}_2(t_{k+1}), \dots, \hat{v}_n(t_{k+1})$ are obtained from the solution of the above linear equations. It

should be noted that the matrix to be inverted in (3.22) is exactly the transpose of the

Jacobian matrix J in (3.21) used to solve the original DNN equations (3.1). Therefore the

LU factors of the Jacobian required to solve the ADNN are just the transpose of the LU

factors used in solving the original DNN. Because the J matrix in (3.21) has been built and decomposed into LU factors during the integration of the original DNN, the solution of ADNN integration (3.22) can be done without re-doing the LU decomposition. In this way, the solution of the adjoint DNN can be achieved at an incremental computation effort once the original DNN is solved.

For example, if we apply trapezoidal rule integration, the particular expressions of J matrix and ADNN linear equations can be obtained by setting $m=0$, $a_0=1$, $b_0=1/2$, $b_{-1}=1/2$ in (3.21) and (3.22). As an example, for a 3rd order DNN and ADNN, (3.21) and (3.22) become

$$J = \begin{bmatrix} I & -\frac{\Delta t}{2}I & 0 \\ 0 & I & -\frac{\Delta t}{2}I \\ -\frac{\Delta t}{2}\frac{\partial f_{ANN}^T}{\partial v_1}(t_{k+1}) & -\frac{\Delta t}{2}\frac{\partial f_{ANN}^T}{\partial v_2}(t_{k+1}) & -\frac{\Delta t}{2}\frac{\partial f_{ANN}^T}{\partial v_3}(t_{k+1}) + I \end{bmatrix} \quad (3.23)$$

$$\begin{bmatrix} I & 0 & -\frac{\Delta t}{2}\frac{\partial f_{ANN}^T}{\partial v_1}(t_{k+1}) \\ -\frac{\Delta t}{2}I & I & -\frac{\Delta t}{2}\frac{\partial f_{ANN}^T}{\partial v_2}(t_{k+1}) \\ 0 & -\frac{\Delta t}{2}I & -\frac{\Delta t}{2}\frac{\partial f_{ANN}^T}{\partial v_3}(t_{k+1}) + I \end{bmatrix} \begin{bmatrix} \hat{v}_1(t_{k+1}) \\ \hat{v}_2(t_{k+1}) \\ \hat{v}_3(t_{k+1}) \end{bmatrix} = \begin{bmatrix} \hat{v}_1(t_k) - \frac{\Delta t}{2}(-\frac{\partial f_{ANN}^T}{\partial v_1}(t_k)\hat{v}_3(t_k) + \frac{\partial f}{\partial y}(t_k) + \frac{\partial f}{\partial y}(t_{k+1})) \\ \hat{v}_2(t_k) + \frac{\Delta t}{2}(\frac{\partial f_{ANN}^T}{\partial v_2}(t_k)\hat{v}_3(t_k) + \hat{v}_1(t_k)) \\ \hat{v}_3(t_k) + \frac{\Delta t}{2}(\frac{\partial f_{ANN}^T}{\partial v_3}(t_k)\hat{v}_3(t_k) + \hat{v}_2(t_k)) \end{bmatrix} \quad (3.24)$$

As can be seen from (3.23) and (3.24), the matrix for solving ADNN equations is the exact transpose of J matrix, and therefore the LU factors in (3.23) can be re-used when solving (3.24).

This computational feature becomes even more beneficial for sensitivity analysis w.r.t. multiple variables. In the DNN model, the number of hidden neurons in f_{ANN} part reflects the degree of non-linearity between DNN dynamic inputs and outputs. More nonlinear problems will require more hidden neurons, therefore more internal weights w have to be trained. The brute force perturbation method for sensitivity analysis of the neural network training error w.r.t. many weight parameters requires solving a perturbed DNN for each weight, leading to massive amount of DNN simulations. Using the proposed technique, only one original DNN and one adjoint DNN need to be solved regardless of the number of variables. In this way, the proposed technique substantially speeds up DNN sensitivity analysis.

3.4.3 Algorithm of ADNN-Enabled Transient Training

Preparation Stage: Define the dynamic inputs and outputs of DNN models, i.e., $u(t)$ and $y(t)$. Generate transient waveform training data $u_d^i(t)$ and $y_d^i(t)$, $i=1,2,\dots,N_T$ from detailed simulation or measurement of original circuit, where N_T is the number of input-output waveform pairs. By supplying different input waveforms to the original circuit (i.e., $u_d(t)$), we obtain different output waveform data (i.e., $y_d(t)$). Optionally, generate time-derivative data of each $u_d^i(t)$ and $y_d^i(t)$ by connecting unit inductor/capacitors and controlled sources to the inputs and outputs of the original circuit.

Initial Training: Using optional data generated from the preparation stage, train the static MLP part, i.e., f_{ANN} , to learn the data of input-output waveforms $u_d(t)$ and $y_d(t)$ and

their time-derivatives up to the n^{th} order derivative. The conventional MLP training techniques such as backpropagation [1] or conjugate gradient algorithms [1] can be used. The result of this training is an approximate f_{ANN} where the model dynamics has not been enforced. This result serves as a good starting point for the transient based training of the dynamic behavior of the DNN next.

Transient Oriented Training: The overall transient oriented DNN training exploiting ADNN sensitivity analysis is summarized in the following steps.

Step 1: Excite the original DNN with the i^{th} input waveform data $u_d^i(t)$, (initially $i=1$). Use numerical integration to solve the original DNN equations forward from time T_1 up to T_2 with user-specified initial condition $v(T_1)$. At each time step, the J matrix is factorized and the resulting LU factors are recorded after being used in integration. Compute the l_2 -based per-waveform training error E_{d_i} using the i^{th} output waveform data $y_d^i(t)$ and the DNN solution $y(t)$, i.e. equation (3.16).

Step 2: Excite the ADNN using the waveform difference between the DNN model output waveform and present training waveform, i.e., equation (3.19). Use numerical integration to solve the ADNN equations backward from time T_2 to T_1 , starting with zero boundary conditions at T_2 . The LU factors stored in Step 1 are recalled periodically for re-use in solving the ADNN.

Step 3: Compute the entire Jacobian matrix $\frac{\partial f_{\text{ANN}}}{\partial w}$ through the MLP backpropagation method [1] (or the static adjoint neural network method [111] if the sensitivity to be computed is w.r.t. the DNN external inputs s instead of internal weights w .) The ADNN

solution in Step 2, i.e., $\hat{y}(t)$, and the Jacobian $\frac{\partial f_{\text{ANN}}}{\partial \mathbf{w}}$ are utilized to calculate the sensitivities of the per-waveform error E_{d_i} of (3.16) with respect to all the weight parameters in $\mathbf{w}$.

Comment: As indicated in the above step, a single ADNN solution is adequate for computing the error sensitivities w.r.t. all DNN internal weights $\mathbf{w}$. This is a significant computational advantage over the perturbation method where one additional simulation of original DNN is required for computing the sensitivity with respect to each weight parameter.

Step 4: If the waveform data used is not the first waveform (i.e., if $i > 1$), then the per-waveform training error E_{d_i} computed in Step 1 and its sensitivities w.r.t. $\mathbf{w}$ computed in Step 4 are added to those of the previous waveform (i.e., waveform $i-1$). Proceed to the next input-output waveform pair, i.e., set $i=i+1$; if $i \leq N_T$, then go to Step 1.

Step 5: Based upon the total training error E_d and its sensitivities accumulated over all the waveforms in the training data, use efficient gradient-based training algorithm (e.g. quasi-Newton method) to calculate the update to weight parameters $\mathbf{w}$.

Step 6: Update the DNN weight parameters $\mathbf{w}$ and continue with Step 1-6 until the total training satisfies user-defined threshold.

Through the above systematic algorithm, the DNN model can be trained to learn the nonlinear transient behavior from the input-output waveforms. The use of adjoint sensitivity technique makes the training process much more efficient than otherwise the use of perturbation technique.

3.4.4 ADNN Based Algorithm for Determining Sensitivity w.r.t.

Static Inputs to DNN

The algorithm presented above can also be extended to cover another case of the proposed ADNN technique, i.e., sensitivity analysis of trained DNN w.r.t. its external static input variables s . Figure 3.2 shows the block schematic of the proposed approach. This sensitivity will be useful for transient optimization of nonlinear circuits using DNN model where s contain optimization variables. In this case, the dynamic input waveform exciting the trained DNN will be a signal in the circuit to be designed. The algorithm for such sensitivity analysis will be the Steps 1-3 described above with the following changes in Steps 1 and 2: In Step 1, the set of input waveform data is replaced by one input waveform $u(t)$ which is the excitation received by trained DNN, and the training error function is replaced by the circuit design objective function. In Step 2, the excitation to the ADNN is replaced by $\frac{\partial f}{\partial y}$, as defined in Equation (3.10).

3.5 Summary

In this chapter, a new ADNN structure is proposed to facilitate efficient adjoint sensitivity computation. The proposed ADNN is a time-varying linear model closely related to the original nonlinear DNN model, i.e. Both share the same number of states, excitations of ADNN are at the corresponding location of DNN outputs and LU factors of DNN Jacobian matrix can be reused in solving ADNN equations. Through the proposed ADNN, the exact sensitivities of DNN energy function with respect to both DNN internal weights and external inputs are derived in a much computationally efficient

manner. Furthermore, we showed how to use these adjoint sensitivities for the purpose of training DNN to learn transient waveform data and performing sensitivity analysis of trained DNN models. In the next chapter, applications of the proposed ADNN technique in transient modeling and nonlinear design are demonstrated through several high-speed interconnect driver modeling and circuit optimization examples.

Figure 3.2: Block schematic of the ADNN algorithm for sensitivity analysis w.r.t. DNN static inputs s . This technique is proposed for circuit design purpose using trained DNN. The DNN is excited by the circuit signal $u(t)$ and evaluated by numerical integration forward in time. The proposed ADNN is evaluated using numerical integration backward in time. The excitation signal to the ADNN is derived from the solutions of the original DNN and the energy function.

Chapter 4

Application Examples of Proposed ADNN Technique

In order to demonstrate the features of the proposed technique presented in last chapter, such as speed, accuracy, and efficiency, the proposed ADNN technique is applied to DNN-modeling of various nonlinear interconnect drivers and gradient-based signal integrity optimization. Firstly the validity of the proposed method is verified by DNN modeling of a CMOS inverter circuit. Then both circuit-based and physics-based high-speed driver models are developed through ADNN-enabled DNN modeling approach and the advantages of using the ADNN are demonstrated through substantial speedup in transient based training of the those buffer models. Furthermore, we illustrate the significance of proposed ADNN technique in signal integrity based optimization. As we will see, the objective function of such signal integrity based optimization is another special form of the energy function E . Though the proposed ADNN technique, the gradient based transient optimization is achieved allowing the physical/geometric parameters as

continuous changeable design parameters but only requiring tiny fraction of the CPU-time taken by physics-based sensitivity computations.

4.1 CMOS Inverter

This example is for validating the proposed ADNN sensitivity technique. A DNN is trained to learn transient behaviors of a CMOS inverter and error derivatives needed during training are computed via the proposed ADNN. The original inverter is a standard configuration including 2 complementary MOSFETs connected with each other [125] as shown in Figure 4.1. Inputs to the DNN model include dynamic input v_{in} (input voltage) and static input W (transistor size). The model output is v_{out} (output voltage). Training waveforms of the inverter are generated using level-49 BSIM3V3 *HSPICE* MOSFET model [126]. A DNN of order $n = 1$ including a 3-layer MLP with 8 hidden neurons is used. Sensitivities of the dynamic training error function w.r.t. DNN parameters (w & W) using the proposed ADNN technique accurately match those from perturbation method as shown in Table 4-I. Figure 4.2 shows the DNN sensitivity w.r.t. both hidden neuron and output neuron weights (i.e., different elements in the weight vector w) under different dynamic input excitations. Excellent agreements have been achieved between the sensitivities obtained from the proposed ADNN and those from perturbation, thus validating the proposed ADNN technique.

Figure 4.1: Schematic of the CMOS inverter circuit of example 4.1 and its corresponding electric variables. Training waveforms of the inverter are generated using level-49 BSIM3V3 *HSPICE* MOSFET model.

TABLE 4-I: Comparison of sensitivity values of CMOS inverter DNN model in example 4.1

DNN Sensitivity	Perturbation Method	Proposed ADNN Technique	Difference (as %)
$\partial E_d / \partial w_{11}^3$	-2.4327e-02	-2.4336e-02	0.036
$\partial E_d / \partial w_{12}^3$	-5.5123e-03	-5.5128e-03	0.009
$\partial E_d / \partial w_{13}^3$	+1.5897e-03	+1.5904e-03	0.044
$\partial E_d / \partial w_{14}^3$	+8.2581e-03	+8.2598e-03	0.021
$\partial E_d / \partial w_{15}^3$	+6.2217e-04	+6.2241e-04	0.038
$\partial E_d / \partial w_{16}^3$	-2.9649e-04	-2.9663e-04	0.047
$\partial E_d / \partial w_{17}^3$	-1.5563e-05	-1.5545e-05	0.116
$\partial E_d / \partial w_{18}^3$	+1.5573e-02	+1.5576e-02	0.019
$\partial E_d / \partial W$	-1.5638e+01	-1.5748e+01	0.703

(a)

(b)

Figure 4.2: Sensitivity of DNN training error w.r.t. different weight parameters in DNN for the CMOS inverter in example 4.1. (a) Sensitivity w.r.t. weights in the output layer of f_{ANN} , and (b) Sensitivity w.r.t. weights in the hidden layer of f_{ANN} . Excellent agreement is observed between the proposed ADNN technique and the perturbation method.

4.2. Circuit-Based High-Speed Interconnect Driver

This practical example shows the utility of the proposed ADNN sensitivity technique in transient modeling. A DNN model of a CMOS inverter driver is developed. Dynamic and static DNN inputs are $\mathbf{u} = [v_{in}, i_{out}]^T$ and $\mathbf{s} = [W]$ respectively, and DNN output is $\mathbf{y} = [v_{out}]^T$. Two sets of training waveforms, namely, signal data and crosstalk data are generated replacing each transistor in the driver circuit by level-49 BSIM3V3 *HSPICE* model. Signal data is collected by supplying the driver with different values for pulse rise-time T_r [0.1ns, 0.3ns, 0.5ns] and pulse amplitude Amp [2.3V, 2.5V, 2.7V], and varying interconnect-length d [2cm, 3cm, 4cm, 5cm] and CMOS channel width W [100 μ m, 240 μ m] (step 20 μ m). Crosstalk data is generated for different pulse amplitudes [-1.0V 3.0V] (step 0.5V) using the above rise-time and W ranges. Time-interval of interest $[T_1, T_2]$ is [0ns, 6ns]. DC values of input signals are used as initial conditions for integrating the original DNN forward.

DNN's of different dynamic orders (n) and with different f_{ANN} are trained using exact error derivatives from the proposed ADNN following the training algorithms presented in Chapter 3.4.3, and the trained model accuracies are shown in Table 4-II. In Table 4-III, the total training time using proposed ADNN sensitivities are compared with that using direct perturbation of DNN weights. As observed, the total training time is reduced significantly from 72.3hours to 6.8hours by exploiting the proposed ADNN technique. A single ADNN evaluation is sufficient for computing all the derivatives for each training waveform pair. After training is finished, we use the trained DNN for VLSI interconnect simulations with coupled transmission lines. Specifically, a trained DNN model of order 2 with 50 hidden neurons is used 3 times as the nonlinear terminations in the coupled transmission line circuit of Figure 4.3 for signal delay and crosstalk analysis

of VLSI interconnect networks. As can be seen in Figure 4.4, excellent agreement is achieved between *HSPICE* simulations of the original interconnect circuit (including coupled transmission lines and the original *HSPICE* CMOS inverters) and our DNN-based interconnect simulations. This example validates the trained DNN as a correct and accurate representation of the original CMOS inverter circuit. The proposed ADNN technique made the development of such accurate model achievable in a practically acceptable training time.

TABLE 4-II: DNN driver model accuracies for different cases in example 4.2

No. of hidden neurons for DNN of order 2	Average test error	Order of DNN with 50 hidden neurons	Average test error
30	1.05%	1	0.38%
50	0.29%	2	0.29%
70	0.71%	3	0.85%

TABLE 4-III: Time for training DNN driver model in example 4.2

No. of hidden neurons for DNN of order 2	Training time with proposed ADNN	Training time with DNN perturbation
30	4.6hrs	33.2hrs
50	6.8hrs	72.3hrs
70	9.5hrs	140.5hrs

Figure 4.3: A coupled 3-conductor interconnect circuit terminated with nonlinear buffers. This circuit is used to generate testing waveforms for testing trained DNN models of nonlinear drivers. Varying parameters in the pulse signal generator can allow us to obtain different sets of testing waveforms.

(a)

(b)

Figure 4.4: (a) Signal propagation waveforms v_{out1} and (b) crosstalks waveforms v_{out2} and v_{out3} of the VLSI interconnect example of Figure 4.3. Excellent agreement is achieved between the responses of DNN trained using the proposed ADNN technique, with those obtained from original HSPICE simulation.

4.3 Physics-Based Multistage Driver

In this example, we demonstrate the relevance of the proposed ADNN technique for physics-based transient design purpose. Physics based modeling and design are important for next generation CAD of high-speed/high-frequency circuits and systems [127]. A $1\mu\text{m}$ 4-stage CMOS driver [128] shown in Figure 4.5 is considered. Training data is obtained using physics-based semiconductor device simulator *MINIMOS-NT* [129]. The driver load is a transmission line with parameters $R=36\Omega/\text{m}$, $L=360\text{nH}/\text{m}$, $C=100\text{pF}/\text{m}$ and $G=0.01\text{S}/\text{m}$, and terminated with a 5pF capacitor. Training waveforms are generated for sets of specific ranges of rise-time T_r , pulse amplitude Amp and interconnect-length d , i.e. different values of T_r [0.25ns, 0.5ns, 0.75ns], Amp [4.5V, 5.0V, 5.5V], and d [0.08m, 0.14m] (step 0.02m). Driver size is also perturbed 50% around the nominal value as indicated in Figure 4.5. Let W_1 be the transistor gate width of 1^{st} -stage buffer. Dynamic and static DNN inputs are $\mathbf{u} = [v_{in}, i_{out}]^T$ and $\mathbf{s} = [W_1]$ respectively, and DNN output is $y(t) = v_{out}(t)$. A DNN structure with dynamic order $n = 1$ and 30 hidden neurons in f_{ANN} , trained using the derivatives from the proposed ADNN technique resulted in a DNN model with average test error of 0.25%. DNN model's prediction of output voltages with independent test waveforms match very well as shown in Figure 4.6, again validating our transient DNN modeling.

We also performed statistical analysis of the signal integrity of the 4-stage driver circuit shown in Figure 4.5 using the accurate DNN model developed through the ADNN-enabled training algorithm. Monte-Carlo analysis of 50 signal integrity output curves with geometrical parameters as statistical design variables is shown in Figure 4.7. The total evaluation time for 50 output curves based on the geometry tolerance around

the nominal design center is 20 seconds using DNN driver model. However, the required time of *MINIMOS-NT* simulation for 50 different geometries is more than 40 hours. This clearly shows the advantages of our DNN driver model, such as speed and accuracy, over the conventional physics model in high-level circuit design.

Figure 4.5: A 4-stage CMOS driver circuit used in example 4.3. DNN models are trained using proposed ADNN technique to learn the transient input-output waveform data generated using physics-based semiconductor simulator such as *MINIMOS-NT*.

Figure 4.6: The output voltage waveforms of the 4-stage CMOS driver under different excitations and driving different transmission line loads. Excellent agreement is achieved between the responses of DNN trained using proposed ADNN technique, with those of *MINIMOS-NT*. The fact that these *MINIMOS-NT* test waveforms are never used in training confirms the validity of DNN model.

Figure 4.7: Monte-Carlo analysis using the DNN model of physics-based high-speed driver for signal integrity. The fast and accurate DNN model is developed using proposed ADNN-enabled training algorithm.

Electrical power is an important criterion for high-speed digital design. We use our ADNN technique for computing sensitivity of average output power of the driver w.r.t. driver-size under transient excitation. To represent the signal power, we choose the energy function E as

$$E = \int_{T_1}^{T_2} f(y) dt = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} v_{out}(t) i_{out}(t) dt \quad (4.1)$$

where $y(t) = v_{out}(t)$ is the solution obtained from the simulation of the original DNN. Let $h(t)$ symbolically represent the impulse response of the transmission line load. In order to define ADNN excitation $\frac{\partial f}{\partial y}$, we symbolically use the convolution relationship

$i_{out}(t) = \int_{-\infty}^{+\infty} v_{out}(\tau) h(t - \tau) d\tau$ and differentiate the energy function in (4.1) to obtain

$$\begin{aligned} \frac{dE}{ds} &= \int_{T_1}^{T_2} \frac{\partial f}{\partial y} \frac{\partial y}{\partial s} dt = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} \frac{\partial(v_{out}(t) i_{out}(t))}{\partial s} dt \\ &= \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} (i_{out}(t) + \tilde{i}_{out}(t)) \frac{\partial v_{out}(t)}{\partial s} dt \end{aligned} \quad (4.2)$$

where $\tilde{i}_{out}(t) = \int_{-\infty}^{+\infty} v_{out}(\tau) h(\tau - t) d\tau$. Therefore, the excitation of ADNN, i.e., $\frac{\partial f}{\partial y}$ in (3.10),

can be set up as

$$\frac{\partial f}{\partial y} = (i_{out}(t) + \tilde{i}_{out}(t)) / (T_2 - T_1) \quad (4.3)$$

There is no need to do numerical convolution for $i_{out}(t)$ and $\tilde{i}_{out}(t)$. $i_{out}(t)$ is obtained by standard transmission line simulation with excitation $v_{out}(t)$, and $\tilde{i}_{out}(t)$ is obtained by exciting the same transmission line with time-reversed version of signal $v_{out}(t)$, i.e.,

$v_{out}(T_2-t)$. Utilizing the proposed algorithm in Chapter 3.4.4, power sensitivities are computed. As shown in Figure 4.8, sensitivities computed with the proposed ADNN accurately match those from physics-based *MINIMOS-NT* perturbations. Total CPU-time for sensitivity analysis taken by the ADNN is 2 seconds as compared to 6254 seconds taken by the *MINIMOS-NT* perturbations, proving the significance of our proposed ADNN technique in nonlinear transient design. The ADNN helps bridging the gap between physics based- and circuit based-CAD, and achieving physics oriented solutions for circuit design at a fast computation speed.

Figure 4.8: The sensitivities of the average output power of the 4-stage CMOS driver evaluated at 3 different driver sizes. The average power was used as the energy function E while computing the sensitivities using the proposed ADNN technique.

4.4. Physics-Based Interconnect Driver Optimization and Design

In this example, we demonstrate the use of another type of energy function in (3.6) and the use of ADNN technique for physics-based time-domain optimization of high speed interconnects, including the physical/geometrical parameters as optimization variables. The 4-stage interconnect driver circuit [128] similar to Figure 4.5 is used in *MINIMOS-NT* to generate training data. Inputs to the DNN model of 4-stage driver include driver input voltage (v_{in}) as dynamic input, the transistor width of 1st-stage buffer (W_1) and scaling coefficient (S) for subsequent stages in the 4-stage buffer as static inputs. The combinations of W_1 and S would represent the different buffer geometrical dimensions at different stages, i.e., $W_1 S^i$ (power i of S) being the transistor width for the $(i+1)^{th}$ -stage buffer. The output of the DNN model is the driver output voltage (v_{out}). The transient training and test waveform data are gathered in *MINIMOS-NT* by varying the transistor widths under different combinations of W_1 and S . Utilizing the ADNN to provide error sensitivities during training process, a DNN driver model with dynamic order $n=1$ and 25 hidden neurons in f_{ANN} is trained. The average test error of the trained DNN is within 0.5%. Figure 4.9 shows the comparison of DNN model outputs with test waveforms gathered from the *MINOMOS-NT* simulation. Excellent match is achieved even though such test waveforms are never used in training, validating the accuracy of the DNN model trained with the proposed ADNN technique. The CPU time taken by ADNN-based training vs. DNN perturbation based training is 45minutes vs. 5.5hours showing the training speed-up by utilizing proposed ADNN technique.

Figure 4.9: The output voltage waveforms of the 4-stage CMOS driver for 3 different geometries: (1) $W_1=1.0\mu\text{m}$ $S=2.5$; (2) $W_1=1.5\mu\text{m}$ $S=4.0$; (3) $W_1=3.0\mu\text{m}$ $S=3.0$. Excellent agreement is achieved between the responses of DNN with those of *MINIMOS-NT* even though these *MINIMOS-NT* test waveforms are never used in training.

The trained DNN driver model is then used in a high-speed VLSI interconnect circuit for time-domain optimization under specifications imposed on transient responses in the interconnect network. A transmission line load with length = 100mm is supplied to the DNN model. The specifications used in our transient optimization are defined as

$$V_{spec}(t) = \begin{cases} 4.75V & \text{as lower specification for } 1ns \leq t \leq 2ns \\ 0.25V & \text{as upper specification for } t \geq 3.5ns \\ 0V & \text{otherwise} \end{cases} \quad (4.4)$$

For transient optimization, error functions can be formulated in terms of design specifications following [122]. We formulate a continuous time-domain error function $e(t)$ as

$$e(t) = v_{out}(t) - V_{spec}(t) \quad (4.5)$$

To facilitate our continuous time-domain optimization, we formulate a continuous-version of one-sided least p^{th} function [103] as

$$H(e(t)) = \left[\int_{\tau_1}^{\tau_2} [g(t)e(t)]^p dt \right]^{1/p} \quad (4.6)$$

where $g(t)$ is a weighting function described by

$$g(t) = \begin{cases} 1 & \text{if } e(t) > 0 \text{ and } V_{spec}(t) \text{ is upper specification} \\ -1 & \text{if } e(t) < 0 \text{ and } V_{spec}(t) \text{ is lower specification} \\ 0 & \text{otherwise} \end{cases} \quad (4.7)$$

We define the objective function for optimization in the form of energy function (3.6) as

$$E = \int_{\tau_1}^{\tau_2} f(y(t)) dt = H^p(e(t)) = \int_{\tau_1}^{\tau_2} (g(t)(v_{out}(t) - V_{spec}(t)))^p dt \quad (4.8)$$

where $y = v_{out}(t)$ is the DNN model output and $[T_1, T_2]$ is defined as $[0\text{ns}, 5\text{ns}]$ in this example.

The optimization problem now is to find feasible W_1 and S such that the objective function E is minimized until all the specifications are satisfied. In order to perform such optimization, we need the sensitivity information of dE/dW_1 and dE/dS , where W_1 and S are the external static inputs to the DNN model (i.e., s in Equation (3.1)). We computed such sensitivities using the proposed ADNN and compared with that from direct *MINIMIS-NT* perturbation as shown in Figures 4.10 and 4.11 for various combinations of W_1 and S . The excellent match between the proposed ADNN sensitivities and that from perturbation confirms the validity of the proposed ADNN technique. Randomly choosing $W_1=1.5\mu\text{m}$ and $S=2.2$ as the starting point and the optimization is carried using derivatives obtained from the proposed ADNN. In this example, the ADNN excitation appearing in (3.10) is described as

$$\frac{\partial f}{\partial y} = p(g(t)(v_{out}(t) - V_{spec}(t))^{p-1} \quad (4.9)$$

In our example, we used $p=3$. During the optimization process, both the original DNN and ADNN are numerically integrated by trapezoidal rule. When solving original DNN, at each time step the factorized $\mathbf{J}$ matrix in the last iteration of Newton-Raphson is saved and reused at the corresponding time point of ADNN integration. Let N_t be the number of time samples per waveform. Upon finishing the original DNN analysis, N_t forward/backward substitutions with the LU -factored $\mathbf{J}^T$ matrix are performed to obtain the ADNN solutions at all time points leading to sensitivity solutions for optimization. After performing the optimization, the optimal values are $W_1=2.8\mu\text{m}$ and $S=3.6$.

Buffer sizing is an important aspect of high-speed VLSI interconnect design. For example, long transmission lines need to be driven by a large driver buffer to maintain signal quality, and short transmission lines can be driven by a small buffer to minimize power. A good buffer model together with transmission line loads will be necessary to effectively perform such design task. Here we use our DNN buffer model trained with physics data to achieve this purpose. In practice, to cope with different interconnect loads, the correct sizes of buffers can be found through optimization process. To illustrate the concept of the proposed ADNN technique for a basic buffer sizing example, we performed a second optimization with the same goal as (4.4) to find out the suitable values of buffer sizes for long interconnect load (with transmission line length of 200mm). Following a similar procedure as the first optimization, the optimal values of design variables are achieved at $W_1=5.7\mu\text{m}$ and $S=3.4$. The transient responses v_{out} before and after optimization together with the *MINIMOS-NT* results are shown in Figure 4.12 and Figure 4.13 for two optimization cases. As can be observed, all the specifications have been satisfied. The improvements of signal quality and reduction in signal delay are obvious. The signal responses of the interconnect circuit simulated with DNN at the optimal buffer sizes are in excellent agreement with that simulated by *MINIMOS-NT*. In addition, the buffer sizes after the second optimization is quite large compared with those from the first optimization, demonstrating the effect of buffer sizing design for different scenarios of transmission line loads.

The ADNN sensitivity enabled effective training of DNN for this transient modeling and optimization example. Using the trained DNN buffer model together with ADNN sensitivity, the buffer-sizing optimization using physics based buffer information is

achieved in 8 seconds. Without the neural network approach, direct optimization by driving *MINIMOS-NT* physics simulator with perturbation derivatives would require 10.5 hours. The proposed ADNN makes it possible to use dynamic neural networks for high-speed interconnect design with physics-level accuracy at only a fraction of time for direct physics based optimization.

Figure 4.10: Sensitivity of the objective function for time-domain signal integrity optimization of high-speed VLSI interconnects (in example 4.4) w.r.t. scaling coefficient s for a subset of W_1 & s parameters used for training (1) $W_1=1.0\mu\text{m}$; (2) $W_1=1.3\mu\text{m}$ and (3); $W_1=1.6\mu\text{m}$. The one-sided least p^{th} function was used to formulate the energy function E while computing the sensitivities using the proposed ADNN technique.

Figure 4.11: Sensitivity of the objective function for time-domain signal integrity optimization of high-speed VLSI interconnects (in example 4.4) w.r.t. 1st stage buffer width for a subset of W_1 & S parameters used for training (1) $S=2.0$; (2) $S=2.1$; and (3) $S=2.2$. The one-sided least p^{th} functions was used to formulate the energy function E while computing the sensitivities using the proposed ADNN technique.

(a)

(b)

Figure 4.12: (a) Responses of the 4-stage CMOS driver models before and after signal integrity optimization. The DNN based optimal solutions are verified by direct physics-based *MINIMOS-NT* simulation. (b) Example of ADNN inputs and outputs used to generate sensitivities for optimization. DNN based optimization utilizing proposed ADNN sensitivity achieves physics-level accuracies at only a tiny fraction of the CPU time required by direct physics based optimization.

(a)

(b)

Figure 4.13: (a) Responses of the 4-stage CMOS driver models before and after signal integrity optimization. The DNN based optimal solutions are verified by direct physics-based *MINIMOS-NT* simulation. (b) Example of ADNN inputs and outputs used to generate sensitivities for optimization. Compared with Figure 4.12 these results are obtained for long transmission line loads, yielding optimal buffer sizing for different interconnects.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In recent years, ANN-based computer-aided design (CAD) approaches have been recognized as fast and accurate tools for microwave modeling, simulation, and optimization. The neural models have been successfully used to replace computationally intensive physics/EM models of active/passive devices to speed up microwave design. As high-speed VLSI interconnect is often connected with IC input and output ports, accurate and fast representation of nonlinear transient behaviors becomes a key step of successful digital CAD including multi-chip modules and multi-layer printed circuit boards (PCB). Aimed at filling the gap between neural network technique and nonlinear transient behavior modeling scenario, this thesis has presented a novel ADNN sensitivity technique to further expand DNN into a more powerful and generic technique. The advantages of the proposed ADNN technique are summarized as follows:

- It provides fast sensitivity computations of DNN energy functions with respect to DNN internal parameters and external inputs. Utilizing the elegant

adjoint concept, computation of all the desired sensitivities only involves one simulation of the original DNN and proposed ADNN.

- It facilitates an efficient training algorithm for DNN to learn nonlinear transient waveform data. It avoids the cumbersome perturbation based DNN sensitivity computations thus dramatically speeds up transient oriented DNN training process. In this way, ADNN opens the door for expanding neural network techniques to a new area of application, namely, transient CAD of nonlinear high-frequency/high-speed circuits.
- It also allows sensitivities of electrical criteria w.r.t. geometrical/physical design parameters of a nonlinear component to be computed efficiently. These features make it possible to use neural networks trained with device physics data in high-speed circuit optimization, achieving physics-level accuracies using only a fraction of the time needed by direct physics-based optimization.

For the first time, this research work successfully has addressed the neuro-based nonlinear transient modeling and design at both circuit level and physics level. It represents a new and optimistic direction of next generation CAD algorithms for high-speed VLSI interconnect design in deep sub-micron technologies.

5.2 Suggestions for Future Research

Neural network approach has been proved as a very powerful and efficient modeling technique for microwave devices and systems. It has a promising future for all stages of

RF/microwave circuits design such as simulation, optimization and statistical design. As DNN technique has been accepted as a fundamental tool to address nonlinear dynamic behavior in circuit simulation environment, what is presented in this thesis is a small progressive step by bridging the gap between the existing DNN and a new research area, i.e. transient modeling and design. It can be expected that more research and applications will appear in the future.

An interesting topic would be the idea of incorporating existing dynamic knowledge of the original nonlinear circuit into the presented transient modeling framework. The presented ADNN-enabled algorithm directly utilizes the external input and output waveforms of the nonlinear circuit for training without referring to any details inside the circuit. In reality, any available knowledge of the original circuit should help to achieve higher accuracy with less effort. Existing knowledge of nonlinear circuit can be collected in the form of extra measurements of system orders and internal states and etc. How to embed such dynamic knowledge into both ADNN and DNN and develop a suitable training scheme will be a new challenging topic for future research.

In this thesis, we have illustrated several examples of high-speed driver modeling and signal integrity optimization. In real design, the actual schematic of driver circuit is much more complicated. To accurately predict the signal transmission in high-speed interconnect network, the performance of receivers is also very critical. As the proposed approach doesn't rely on the internal details of the nonlinear circuit, in the future it can be expected the proposed ADNN technique will find more applications in the transient modeling and design of actual complex high-speed drivers and receivers.

Since the nonlinear dynamic circuit to be modeled is a stable system, stability of the developed model should be satisfied in order to use the model for simulation and design. To achieve that, some constraints have to be imposed during the DNN training using proposed ADNN-enabled algorithm. How to formulate such constraints in terms of stability criterion and implement them during ADNN-enabled training process will be an optimistic direction to further enhance the proposed technique.

Bibliography

- [1] Q.J. Zhang and K.C. Gupta, *Neural Networks for RF and Microwave Design*. Norwood, MA: Artech House, 2000.
- [2] Q.J. Zhang, K.C. Gupta, and V.K. Devabhaktuni, "Artificial neural networks for RF and microwave design: From theory to practice," *IEEE Trans. Microwave Theory Tech.* vol. 51, pp. 1339-1350, 2003.
- [3] Y. Fang, M.C.E. Yagoub, F. Wang, and Q.J. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," *IEEE Trans. Microwave Theory Tech.*, vol. 48, pp. 2335-2344, 2000.
- [4] J.J. Xu, M. C.E. Yagoub, R.T. Ding, and Q.J. Zhang, "Neural based dynamic modeling of nonlinear microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 50, pp. 2769-2780, 2002.
- [5] Y. Cao, J.J. Xu, V.K. Devabhaktuni, R.T. Ding, and Q.J. Zhang, "An adjoint dynamic neural network technique for exact sensitivities in nonlinear transient modeling and high-speed interconnect design," *IEEE MTT-S Int. Microwave Symp. Dig.*, Philadelphia, PA, 2003 (accepted).
- [6] P. Burrascano, S. Fiori, and M. Mongiardo, "A review of artificial neural networks applications in microwave computer-aided design", *Int. J. RF and Microwave CAE*, vol. 9, pp. 158-174, 1999.

- [7] F. Wang and Q.J. Zhang, "Knowledge-based neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2333-2343, 1997.
- [8] F. Wang, V.K. Devabhaktuni, and Q.J. Zhang, "A hierarchical neural network approach to the development of a library of neural models for microwave design," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 2391-2403, 1998.
- [9] A. Patnaik, R.K. Mishra, G.K. Patra, and S.K. Dash, "An artificial neural network model for effective dielectric constant of microstripline," *IEEE Trans. Antennas Propagat.*, vol. 45, p. 1697, 1997.
- [10] G. Antonini and A. Orlandi, "Gradient evaluation for neural-networks-based electromagnetic optimization procedures," *IEEE Trans. Microwave Theory Tech.*, vol. 48, pp. 874-876, 2000.
- [11] Q.J. Zhang, F. Wang, and M.S. Nakhla, "Optimization of high-speed VLSI interconnects: A review," *Int. J. Microwave Millimeter-Wave CAE*, vol. 7, pp. 83-107, 1997.
- [12] T. Horng, C. Wang, and N.G. Alexopoulos, "Microstrip circuit design using neural networks," *IEEE MTT-S Int. Microwave Symp. Dig.*, Atlanta, GA, 1993, pp. 413-416.
- [13] V.K. Devabhaktuni and Q.J. Zhang, "Neural network training-driven adaptive sampling algorithm for microwave modeling," *Proc. 30th European Microwave Conf.*, Paris, France, Oct. 2000, pp. 222-225.
- [14] R.L. Mahajan, "Design and optimization through physical/neural network models," *IEEE MTT-S Int. Microwave Symp. Workshop on Appl. of ANN to Microwave Design*, Denver, CO, June 1997, pp. 1-16.

- [15] Q.J. Zhang, "A neural network paradigm for microwave modeling, simulation and optimization," *IEEE MTT-S Int. Microwave Symp. Workshop on Appl. of ANN to Microwave Design*, Denver, CO, June 1997, pp. 87-107.
- [16] P.M. Watson, M.Y. Mah, and L.L. Liou, "Input variable space reduction using dimensional analysis for artificial neural network modeling," *IEEE MTT-S Int. Microwave Symp. Dig.*, Anaheim, CA, June 1999, pp. 269-272.
- [17] F. Wang and Q.J. Zhang, "Incorporating functional knowledge into neural networks," *Proc. IEEE Int. Conf. Neural Networks*, Houston, TX, June 1997, pp. 266-269.
- [18] M. Nakhla, "Application of neural networks to simulation and optimization of high-speed interconnects," *IEEE MTT-S Int. Microwave Symp. Workshop on Appl. of ANN to Microwave Design*, Denver, CO, June 1997, pp. 49-66.
- [19] A.H. Zaabab, Q.J. Zhang, and M.S. Nakhla, "Device and circuit level modeling using neural networks with faster training based on network sparsity," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 1696-1704, 1997.
- [20] A. Veluswami, M.S. Nakhla, and Q.J. Zhang, "The application of neural networks to EM-based simulation and optimization of interconnects in high-speed VLSI circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 712-723, 1997.
- [21] J. Bandler, M. Ismail, J. Rayas-Sanchez, and Q. Zhang, "New directions in model development for RF/microwave components utilizing artificial neural networks and space mapping," *IEEE APS Int. Symp. Dig.*, Orlando, FL, July 1999, pp. 2572-2575.

- [22] J.W. Bandler, M.A. Ismail, J.E. Rayas-Sanchez, and Q.J. Zhang, "Neuromodeling of microwave circuits exploiting space mapping technology," *IEEE MTT-S Int. Microwave Symp. Dig.*, Anaheim, CA, June 1999, pp. 149-152.
- [23] P.M. Watson and K.C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 44, pp. 2495-2503, 1996.
- [24] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Development of knowledge based artificial neural network models for microwave components," *IEEE MTT-S Int. Microwave Symp. Dig.*, Baltimore, MD, June 1998, pp. 9-12.
- [25] P.M. Watson, K.C. Gupta, and R.L. Mahajan, "Applications of knowledge-based artificial neural network modeling to microwave components," *Int. J. RF and Microwave CAE*, vol. 9, pp. 254-260, May 1999.
- [26] K.C. Gupta, "EM-ANN models for microwave and millimeter-wave components," *IEEE MTT-S Int. Microwave Symp. Workshop on Appl. of ANN to Microwave Design*, Denver, CO, June 1997, pp. 17-47.
- [27] P. Watson and K.C. Gupta, "EM-ANN models for via interconnects in microstrip circuits," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Francisco, CA, June 1996, pp. 1819-1822.
- [28] P.M. Watson and K.C. Gupta, "Design and optimization of CPW circuits using EM-ANN models for CPW components," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 2515-2523, 1997.

- [29] P. Watson, G. Creech, and K. Gupta, "Knowledge based EM-ANN models for the design of wide bandwidth CPW patch/slot antennas," *IEEE APS Int. Symp. Dig.*, Orlando, FL, July 1999, pp. 2588-2591.
- [30] G.L. Creech, B. Paul, C. Lesniak, T. Jenkins, R. Lee, and M. Calcaterra, "Artificial neural networks for accurate microwave CAD applications," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Francisco, CA, June 1996, pp. 733-736.
- [31] G.L. Creech, B.J. Paul, C.D. Lesniak, T.J. Jenkins, and M.C. Calcaterra, "Artificial neural networks for fast and accurate EM-CAD of microwave circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 794-802, 1997.
- [32] J.W. Bandler, M.A. Ismail, J.E. Rayas-Sánchez and Q. J. Zhang, "Neuromodeling of microwave circuits exploiting space mapping technology," *IEEE Trans. Microwave Theory Tech.*, vol. 47, , pp. 2417-2427, 1999.
- [33] D. Schreurs, J. Verspecht, S. Vandenberghe, and E. Vandamme, "Straightforward and accurate nonlinear device model parameter-estimation method based on vectorial large-signal measurements," *IEEE Trans. Microwave Theory Tech.*, vol. 50, pp. 2315-2319, 2002.
- [34] Y. Harkouss, J. Rousset, H. Chehade, E. Ngoya, D. Barataud, and J.P. Teyssier, "Modeling microwave devices and circuits for telecommunications system design," *Proc. IEEE Int. Conf. Neural Networks*, Anchorage, Alaska, May 1998, pp. 128-133.

- [35] A.H. Zaabab, Q.J. Zhang, and M. Nakhla, "Analysis and optimization of microwave circuits & devices using neural network models," *IEEE MTT-S Int. Microwave Symp. Dig.*, San Diego, CA, May 1994, pp. 393-396.
- [36] V.K. Devabhaktuni, C. Xi, F. Wang, and Q.J. Zhang, "Robust training of microwave neural models," *IEEE MTT-S Int. Microwave Symp. Dig.*, Anaheim, CA, June 1999, pp. 145-148.
- [37] G. Kothapalli, "Artificial neural networks as aids in circuit design," *Microelectronics J.*, vol. 26, pp. 569-578, 1995.
- [38] V.B. Litovski, J.I. Radjenovic, Z.M. Mrcarica, and S.L. Milenkovic, "MOS transistor modeling using neural network," *Elect. Lett.*, vol. 28, pp. 1766-1768, 1992.
- [39] V.K. Devabhaktuni, C. Xi, and Q.J. Zhang, "A neural network approach to the modeling of heterojunction bipolar transistors from S-parameter data," *Proc. 28th European Microwave Conf.*, Amsterdam, Netherlands, Oct. 1998, pp. 306-311.
- [40] M. Vai and S. Prasad, "Qualitative modeling heterojunction bipolar transistors for optimization: A neural network approach," *Proc. IEEE/Cornell Conf. Adv. Concepts in High Speed Semiconductor Dev. and Circuits.*, 1993, pp. 219-227.
- [41] K. Shirakawa, M. Shimiz, N. Okubo, and Y. Daido, "A large-signal characterization of an HEMT using a multilayered network," *IEEE Trans. Microwave Theory Tech.*, vol. 45, pp. 1630-1633, 1997.

- [42] K. Shirakawa, M. Shimizu, N. Okubo, and Y. Daido, "Structural determination of multilayered large-signal neural-network HEMT model," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 1367-1375, 1998.
- [43] G. Fedi, S. Manetti, G. Pelosi, and S. Selleri, "Design of cylindrical posts in rectangular waveguide by neural network approach," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, July 2000, pp. 1054-1057.
- [44] A.H. Zaabab, Q.J. Zhang, and M. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1349-1358, 1995.
- [45] M. Vai, S. Wu, B. Li, and S. Prasad, "Creating neural network based microwave circuit models for analysis and synthesis," *Proc. Asia Pacific Microwave Conf.*, Hong Kong, Dec. 1997, pp. 853-856.
- [46] Y. Harkouss, J. Rousset, H. Chehade, E. Ngoya, D. Barataud, and J.P. Teyssier, "The use of artificial neural networks in nonlinear microwave devices and circuits modeling: An application to telecommunication system design," *Int. J. RF and Microwave CAE*, vol. 9, pp. 198-215, May 1999.
- [47] V. Rizzoli, A. Neri, D. Masotti, and A. Lipparin, "A new family of neural network-based bi-directional and dispersive behavioral models for nonlinear RF/Microwave subsystems," *Int. J. RF and Microwave CAE*, vol. 12, pp. 51-70, 2002.
- [48] G. Fedi, R. Giomi, M. Giustini, R. Nesti, G. Pelosi, and S. Selleri, "An artificial neural network for the analysis of profiled corrugated circular horns," *IEEE APS Int. Symp. Dig.*, Orlando, FL, July 1999, pp. 2604-2607.

- [49] C. Christodoulou, A. El Zooghby, and M. Georgiopoulos, "Neural network processing for adaptive array antennas," *IEEE APS Int. Symp. Dig.*, Orlando, FL, July 1999, pp. 2584-2587.
- [50] R. Mishra and A. Patnaik, "Neurospectral analysis of coaxial fed rectangular patch antenna," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, July 2000, pp. 1062-1065.
- [51] G. Washington, "Aperture antenna shape prediction by feedforward neural networks," *IEEE Trans. Antennas Propagat.*, vol. 45, pp. 683-688, 1997.
- [52] R.K. Mishra and A. Patnaik, "Neural network based CAD model for the design of square-patch antennas," *IEEE Trans. Antennas Propagat.*, vol. 46, pp. 1890-1891, 1998.
- [53] A.H. El Zooghby, C.G. Christodoulou, and M. Georgiopoulos, "Neural network-based adaptive beamforming for one- and two-dimensional antenna arrays," *IEEE Trans. Antennas Propagat.*, vol. 46, pp. 1891-1893, 1998.
- [54] E. Charpentier and J.J. Laurin, "An implementation of a direction-finding antenna for mobile communications using a neural network," *IEEE Trans. Antennas Propagat.*, vol. 47, pp. 1152-1159, 1999.
- [55] H. Yang, C. He, W. Song, and H. Zhu, "Using artificial neural network approach to predict rain attenuation on earth-space path," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, July 2000, pp. 1058-1061.
- [56] J. Lee and A.Lai, "Function-based and physics-based hybrid modular neural network for radio wave propagation modeling," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, July 2000, pp. 446-449.

- [57] M. Vai and S. Prasad, "Automatic impedance matching with a neural network," *IEEE Microwave Guided Wave Lett.*, vol. 3, pp. 353-354, 1993.
- [58] M. Vai and S. Prasad, "Microwave circuit analysis and design by a massively distributed computing network," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 1087-1094, 1995.
- [59] M. Vai and S. Prasad, "Applications of neural networks optimized by the genetic algorithm to microwave systems," *IEEE APS Int. Symp. Dig.*, Orlando, FL, July 1999, pp. 2580-2583.
- [60] E. Bermiani, S. Caorsi, and M. Raffetto, "Time domain inverse scattering by means of neural networks," *IEEE APS Int. Symp. Dig.*, Salt Lake City, UT, July 2000, pp. 1760-1763.
- [61] M. Vai, S. Wu, B. Li, and S. Prasad, "Reverse modeling of microwave circuits with bidirectional neural network models," *IEEE Trans. Microwave Theory Tech.*, vol. 46, pp. 1492-1494, 1998.
- [62] M. Vai and S. Prasad, "Neural networks in microwave circuit design - Beyond black box models," *Int. J. RF and Microwave CAE*, vol. 9, pp. 187-197, May 1999.
- [63] M.H. Bakr, J.W. Bandler, M.A. Ismail, J.E. Rayas-Sanchez, and Q.J. Zhang, "Neural space mapping EM optimization of microwave structures" *IEEE MTT-S Int. Microwave Symp. Dig.*, Boston, MA, June 2000, pp. 879-882.
- [64] P.M. Watson, C. Cho, and K.C. Gupta, "Electromagnetic-artificial neural network model for synthesis of physical dimensions for multilayer asymmetric coupled

- transmission structures,” *Int. J. RF and Microwave CAE*, vol. 9, pp. 175-186, May 1999.
- [65] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume I, pp. 318-362, MIT Press, Cambridge, Massachusetts, 1986.
  - [66] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671-680, 1983.
  - [67] A. J. F. van Rooij, L. C. Jain, R. P. Johnson, *Neural Network Training Using Genetic Algorithms*, World Scientific, 1996.
  - [68] C. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Math. Control Signals Systems*, vol. 2, pp. 303-314, 1989.
  - [69] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, pp. 359-366, 1989.
  - [70] T. Y. Kwok and D. Y. Yeung, “Constructive algorithms for structure learning in feedforward neural networks for regression problems,” *IEEE Trans. Neural Networks*, vol. 8, pp. 630-645, 1997.
  - [71] R. Reed, “Pruning algorithms – a survey,” *IEEE Trans. Neural Networks*, vol. 4, pp. 740-747, September 1993.
  - [72] J. de Villiers and E. Barnard, “Backpropagation neural nets with one and two hidden layers,” *IEEE Trans. Neural Networks*, vol. 4, pp. 136-141, 1992.

- [73] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layer versus three," *IEEE Trans. Neural Networks*, vol. 8, pp. 251-255, 1997.
- [74] C.S. Walker, *Capacitance, Inductance and Crosstalk Analysis*. Boston, MA: Artech House, 1990.
- [75] H. Leung and S. Haykin, "Rational function neural network," *Neural Computation*, vol. 5, pp. 928-938, 1993.
- [76] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4-27, 1990.
- [77] J. Aweya, Q. J. Zhang, and D. Y. Montuno, "A direct adaptive neural controller for flow control in computer networks," in *Proc. IEEE Int. Neural Networks Conf.*, Anchorage, Alaska, pp. 140-145, May 1998.
- [78] S. Haykin, *Neural Networks: A comprehensive foundation*, second edition, Prentice Hall, Upper Saddle River, NJ, 1999.
- [79] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York, NY: Van Nostrand Reinhold, 1993.
- [80] *ADS-Advanced Design System Version 2002*, Agilent Technologies, Santa Rosa, CA, 2002.
- [81] G. Casinovi and A. Sangiovanni-Vincentelli, "A macromodeling algorithm for analog circuits," *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 150-160, 1991.

- [82] K.S. Kundert, G.B. Sorkin, and A. Sangiovanni-vincentelli, "Applying harmonic balance to almost-periodic circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 366-378, 1988.
- [83] R. Battiti, "First and second-order methods for learning: between steepest descent and Newton's method," *Neural Computation*, vol. 4, pp. 141-166, 1992.
- [84] K. Ochiai, N. Toda, and S. Usui, "Kick-out learning algorithm to reduce the oscillation of weights," *Neural Networks*, vol. 7, pp. 797-807, 1994.
- [85] S. J. Perantonis and D. A. Karras, "An efficient constrained learning algorithm with momentum acceleration," *Neural Networks*, vol. 8, pp. 237-249, 1995.
- [86] *Neural Network Toolbox, For Use with Matlab*, The MathWorks Inc., Natick, Massachusetts, 1993.
- [87] R. A. Jacobs, "Increased rate of convergence through learning rate adaptation," *Neural Networks*, vol. 1, pp. 295-307, 1988.
- [88] M. Arisawa and J. Watada, "Enhanced back-propagation learning and its application to business evaluation," *In Proc. IEEE Intl. Conf. Neural Networks*, vol. I, pp. 155-160, Orlando, Florida, July 1994.
- [89] N. Baba, Y. Mogami, M. Kohzaki, Y. Shiraishi, and Y. Yoshida, "A hybrid algorithm for finding the global minimum of error function of neural networks and its applications," *Neural Networks*, vol. 7, pp. 1253-1265, 1994.
- [90] P. P. Van-Der-Smagt, "Minimization methods for training feedforward neural networks," *Neural Networks*, vol. 7, pp. 1-11, 1994.

- [91] R. L. Watrous, "Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization," In *Proc. IEEE First Intl. Conf. Neural Networks*, vol. II, pp. 619-627, San Diego, California, 1987.
- [92] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge, Massachusetts: Cambridge University Press, 1986.
- [93] M. F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525-533, 1993.
- [94] T. R. Cuthbert(Jr.), *Optimization using Personal Computers, chapter Quasi-Newton Methods and Constraints*, pp. 233-314. John Wiley and Sons, New York, NY, 1987.
- [95] P. Gill, W. Murray and M. Wright, *Practical Optimization*, Academic Press, London, England, 1981.
- [96] K. R. Nakano, "Partial BFGS update and efficient step-length calculation for three layer neural network," *Neural Computation*, vol. 9, pp. 123-141, 1997.
- [97] S. Mcloone and G. W. Irwin, "Fast parallel off-line training of multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 8, pp. 646-653, 1997.
- [98] T. Rognvaldsson, "On Langevin updating in multilayer perceptron," *Neural Computation*, vol. 6, pp. 916-926, 1994.
- [99] R. Brunelli, "Training neural nets through stochastic minimization," *Neural Networks*, vol. 7, pp. 1405-1412, 1994.
- [100] S.W. Director and R.A. Rohrer, "The generalized adjoint network and network sensitivities," *IEEE Trans. Circuit Theory*, vol. 16, pp. 318-323, 1969.

- [101] P. Penfield, Jr., R. Stence, and S. Duinker, *Tellegen's Theorem and Electrical Networks*. Cambridge, MA: M.I.T. Press, 1970.
- [102] S.W. Director, "Survey of circuit-oriented optimization techniques," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 3-10, 1971.
- [103] J.W. Bandler, S.H. Chen, "Circuit optimization: the state of the art", *IEEE Trans. Microwave Theory Tech.* vol. 36, pp. 424-443, 1988.
- [104] S.W. Director and R.A. Rohrer, "Automated network design – the frequency-domain case," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 330-337, 1969.
- [105] J.W. Bandler and R.E. Seviara, "Computation of sensitivities for noncommensurate networks," *IEEE Trans. Circuit Theory*, vol. CT-18, pp. 174-177, 1971.
- [106] M.N. S. Swamy, C. Bhushan, and K. Thulasiraman, "Sensitivity invariants for nonlinear networks," *IEEE Trans. Circuit Theory*, vol. CT-19, pp. 599-606, 1972.
- [107] L.O. Chua and P.M. Lin, *Computer-Aided Analysis of Electronic Circuits*. Englewood Cliffs, NJ: Prentice Hall, 1975.
- [108] J.W. Bandler, Q.J. Zhang, and R.M. Biernacki, "A unified theory for frequency-domain simulation and sensitivity analysis of linear and nonlinear circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 1661-1669, 1988.
- [109] M. Mongiardo and R. Ravanelli, "Automated design of corrugated feeds by the adjoint network method," *IEEE Trans. Microwave Theory Tech.* vol. 45, pp. 787-793, 1997.
- [110] N.K. Georgieva, S. Glavic, M.H. Bakr and J.W. Bandler, "Feasible adjoint sensitivity technique for EM design optimization," *IEEE Trans. Microwave Theory Tech.* vol. 50, 2002, pp. 2751-2758.

- [111] J.J. Xu, M.C.E. Yagoub, R.T. Ding, and Q.J. Zhang, "Exact adjoint sensitivity analysis for neural based microwave modeling and design," *IEEE Trans. Microwave Theory Tech.* vol. 51, pp. 226-237, 2003.
- [112] J.J. Xu, M. Yagoub, R.T. Ding, and Q.J. Zhang, "Neural based dynamic modeling of nonlinear microwave circuits," *IEEE MTT-S Int. Microwave Symp. Dig.*, Seattle, WA, 2002, pp. 1101-1104.
- [113] V. Rizzoli and A. Neri, "State of the art and present trends in nonlinear microwave CAD techniques," *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 343-365, 1988.
- [114] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Computation*, vol. 1, pp. 263-269, 1989.
- [115] T. Hryces, *Neurocontrol, Towards An Industrial Control Methodology*, New York: Wiley- Interscience, 1997.
- [116] D.E. Root, J. Wood, N. Tufillaro, D. Schreurs, and A. Pekker, "Systematic behavioral modeling of nonlinear microwave/RF circuits in the time domain using techniques from nonlinear dynamical systems," *International Behavioral Modeling and Simulation Workshop*, Santa Rosa, CA, Nov. 2002.
- [117] R. Achar and M.S. Nakhla "Simulation of high-speed interconnects," *Proc. IEEE*, vol. 89, pp. 693-728, 2001.
- [118] M.S. Nakhla and Q.J. Zhang Ed, *Modeling and Simulation of High-Speed VLSI Interconnects*, Kluwer Academic Publishers, Boston, MA, 1994.

- [119] P. Gunupudi and M. Nakhla, "Nonlinear circuit-reduction of high-speed interconnect networks using congruent transformation techniques," *IEEE Trans on Advanced Packaging*, vol. 24, no. 3, pp. 317-325, 2001.
- [120] A. Dounavis, R. Achar, and M.S. Nakhla, "Efficient sensitivity analysis of lossy multiconductor transmission lines with nonlinear terminations," *IEEE Trans. Microwave Theory Tech.*, vol. 49, pp. 2292-2299, 2001.
- [121] S. Lum, M. Nakhla and Q.J. Zhang, "Sensitivity analysis of lossy coupled transmission lines with nonlinear terminations", *IEEE Trans. Microwave Theory Tech.* vol. 42, pp. 607-615, 1994.
- [122] Q.J. Zhang, S. Lum, and M. Nakhla, " Minimization of delay and crosstalk in high-speed VLSI interconnects", *IEEE Trans. Microwave Theory Tech.* vol. 40, pp. 1555-1563, 1992.
- [123] J. Cheng, A.C. Cangellaris, A.C. Yaghmour, and J.L. Prince, "Sensitivity analysis of multiconductor transmission lines and optimization for high-speed interconnect circuit design," *IEEE Trans on Advanced Packaging*, vol. 23, pp. 132-141, 2000.
- [124] I.S. Stievano, Z. Chen, D. Becker, F.G. Canavero, G. Katopis and I.A. Maio, "Behavioral modeling of digital IC input and output ports," *IEEE Topical Meeting of Electrical Performance of Electronic Packaging*, pp. 331 -334, 2001.
- [125] A.S. Sedra and K.C. Smith, *Microelectronic Circuits*, 4th Edition, New York: Oxford University Press, 1998.
- [126] *HSPICE v. 2001.4*, Avant! Corporation, Fremont, CA, USA.

- [127] M.B. Steer, J.W. Bandler, and C.M. Snowden, "Computer aided design of RF and microwave circuits and systems," *IEEE Trans. Microwave Theory Tech.*, vol. 50, pp. 996-1005, 2002.
- [128] J.M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Upper Saddle River, NJ: Prentice Hall, 1996.
- [129] *MINIMOS-NT* v.2.0, Institute for Microelectronics, Technical University Vienna, Vienna, Austria.