Modeling of PN Junction Diode Using Neural Networks and Genetic Algorithm

نمذجة ثنائي أشباه الموصلات باستخدام الشبكات العصبية الصناعية والخوارزم الجيني

Mohamed A. Yakout, Abdelfattah I. Abdelfattah and Ayman S. Elbazz
Electronics & Communications Department, Faculty of Engineering,
Mansoura University, Al- Mansoura, EGYPT

الملخص العربي-يقدم البحث طريقة لنمذجة خواص الوصلة الثنائية (سالب-موجب) باستخدام الشبكات العصبية الصناعية. حيث تم تعلم الشبكة بتطبيق طريقة الانتشار الخلفي ثم الخوارزم الجيني للحصول على تعلم أسرع وأكثر دقة.

Abstract – **The trend of modeling eleetronic devices using neural network is a new research area. The use of that neural network model to analyze electronic circuits is a promising new teehnique in the filed of circuits analysis. The availability of modeling PN junction diode has been earried out using neural networks. The introduced model covers the operation of the PN junction in forward bias over a wide range of temperatures. The neural nctwork is used to model the PN junction employing the genetic algorithm as a learning tool. The proposed technique is faster due to the use of parallel operation from both neural networks and genetic algorithm. Moreover, its results are very accurate.**

I. Introduction

The modeling of electronic devices is considered the key of analysis and solving electronic circuits.

Recently, neural networks have become a much important tool in the signal processing area for speech processing, vision, control system. and more [1]-[4]. They enjoy some distinguished characteristics including the ability to learn from data. to generalize patterns in data, and to model nonlinear relationships. These appealing features make neural networks a good candidate for overcoming some of the difficulties in traditional device and circuit modeling and optimization [5]. However, the neural network learning algorithm; backpropagation, the most famous learning algorithm, needs a bounded, monotonic, nondecreasing, continuous and derivative function in order to work correctly [6].

On the other hand, the genetic algorithm has several advantages. It does not require any continuity or derivativity. It can search the sample space carefully and can therefore more easily avoid local optima. Moreover, it provides a parallel search with a population model.

This paper is concerned with modeling of the PN junction diode in the forward direction. Neural network and genetic algorithm are employed to get the required model covering the temperature range from $-50°C$ to $+150°C$. The proposed approach is much faster and accurate than the traditional one.

Accepted November 3, 1999

II. Artificial Neural Networks

Artificial neural networks (ANNs) are composed of many nonlinear, highly interconnected computational elements operating in parallel. Unlike the serial computers that execute many instructions of certain programs and need memories to store these instructions and sometimes the output (input) data. neural networks respond in parallel to the input presented to them. The result is not stored in a memory, but consists of the overall state of the network after it has reached some equilibrium condition. This means that the information is stored in both the ways processing elements are connected, *i. e.*, the architecture of neural networks and the weighting values of inputs to the processing elements [4].

Fig.1 shows the simplest form of the neuron model [2], [4], [7] and [8]. This model computes a weighted sum of its input from other neurons and produces an output only if this sum exceeds an internal threshold value, μ.

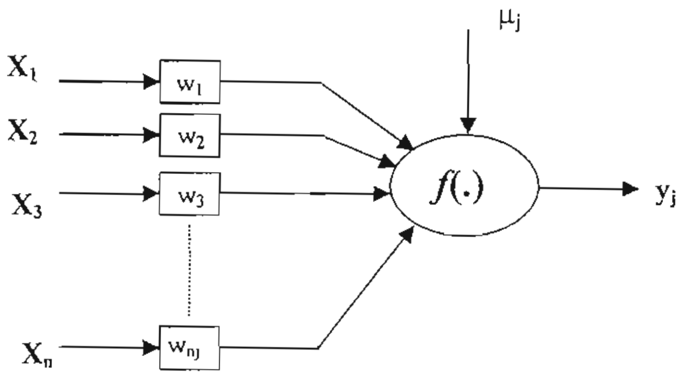

Fig. 1 Neuron Model for ANNs.

Each input x_i is usually weighted by a corresponding weight w_{ij} that represents the strength of the synapse connecting the ith input to the jth output. Weights can be positive or negative values corresponding to excitatory or inhibitory synapses; respectively. The following equation represents this model in mathematical form,

$$y_j = f\left(\sum_{i=1}^{n} x_i w_{ij} - \mu_j\right). \tag{1}$$

Where y_j is the output of the model neuron, μ_j is the threshold value of the neuron j and f(.) is the activation function. Typical activation functions used in ANNs are shown in Fig.2, [2] and [8]. The step and sigmoid function being the more popular types used to date.

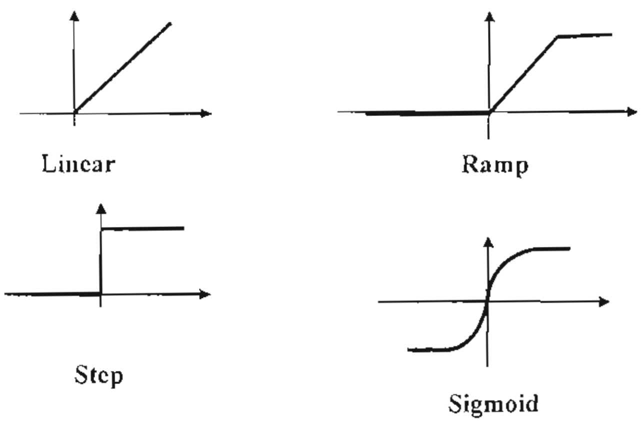

Fig. 2 Typical Activation Functions

III. Genetic Algorithms

Genetic algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics [9]. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form search algorithm with some of the innovative flair of human research. In every generation. a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. The genetic algorithm is an example of a search procedure that uses random choice as a tool to guide a highly exploitative search through a coding of a parameter space.

Genetic algorithms are different from most normal optimization and search procedure in four ways:

(1) *GAs work with a coding of the parameter set, not the parameters themselves*: As an example. consider the optimization problem $f(x)=x^2$ on the integer interval [0, 31], then we can use five bits to code all the integer values in the interval [0, 31].

(2) *GAs search from a population of points, not a single point*: many optimization methods move from a single point in the decision space to the next using some transition rule to determine the next point. This method can lead to false peaks in multimodal (many-peaked) search space. By contrast, GAs work from a rich database of points simultaneously. A population of strings climbing many peaks in parallel: thus the probability of finding a false peak is reduced.

(3) *GAs use payoff (objective function)*: many search techniques require much auxiliary information in order to work properly. For example, gradient techniques need derivatives to be able to climb the current peak. GAs have no need for all these auxiliary information. To perform an effective search they only require payoff values (objective function values) associated with individual strings.

(4) *GAs use probabilistic transition rules, not deterministic rules*: unlike many methods, they use random choice as a tool to guide a search toward regions of the search space with likely improvement.

A simple genetic algorithm that yields good results in many practical problems is composed of the following operators:

(1) The problem to be addressed is defined and captured in an objective function that indicates the fitness of any potential solution.

(2) A population of candidate solutions is initialized subject to certain constraints. Typically, each trial solution is coded as a vector termed a chromosome, with elements being described as genes.

(3) *Reproduction*: it is a process in which individual strings are copied according to their objective function values. Copying strings according to their fitness values means that strings with higher values have higher probability of contributing one or more offspring in the next generation. The reproduction operator may be implemented in algorithm form in a number of ways. Perhaps the easiest is to create a biased roulette wheel where each current string in the population has a roulette wheel slot sized in proportion to its fitness. To reproduce, we simply spin the weighted roulette wheel. In this way, more highly fit strings have a higher number of offspring in the succeeding generation. Once a string has been selected for reproduction, an exact replica of the string is made. This string is then entered into a mating pool, a tentative new population, for further genetic operator action.

(4) *Crossover*: it is applied to two chromosomes (parents) and creates two new chromosomes (offspring) by selecting a random position along the coding and splicing the section that appears before the selected position in the first string with the section that appears after the selected position in the second string, and vice versa.

(5) *Mutation*: although reproduction and crossover produce many new strings, they do not introduce any new information into the population at the bit level. As a source of new bits, mutation is introduced and is applied with a low probability. It inverts a randomly chosen bit on a string.

IV. Diode Forward V-I Characteristics

The current I of a PN junction diode under forward and reverse bias conditions can be approximated by:

$$I = I_o(e^{v/\eta V_T} - 1) \qquad (2)$$

where

v is the forward applied voltage to the diode,
η is a constant (equals to 2 for silicon) and,
V_T is the thermal voltage, given by

$$V_T = \frac{kT}{q} \qquad (3)$$

where

k is the Boltzmann constant = 1.38×10^{-23} joule/K,
T is the temperature in K, and
q is the magnitude of electronic charge.

The reverse saturation current I_0 can be written as:

$$I_o = qAn_i^2\left(\frac{L_p}{\tau_p N_d} + \frac{L_n}{\tau_n N_a}\right) \tag{4}$$

where

A is the cross section area of the junction.

L_p is the diffusion length of the holes.

L_n is the diffusion length of the electrons.

N_d is the doping concentration of the donor atoms in the N side.

N_a is the doping concentration of the acceptor atom in the P side,

τ_p is the hole life time in the N region, and

τ_n is the electron life time in the P region.

The parameters within the parentheses are relatively insensitive to temperature variation [10], while the intrinsic concentration of carriers n_i is realted to the temperature T as follow [11]:

$$n_i = BT^{3/2}e^{-(W_g/2kT)} \tag{5}$$

where

W_g is the forbidden band or energy gap (the energy required to break a covalent band) in joules and

B is a constant.

The energy gap W_g is a linear function of temperature over the useful temperature range. Thus W_g can be written as

$$W_g = W_{go}(1 - CT) \tag{6}$$

where

C is the gap-energy temperature coefficient ($2.8\text{x}10^{-4}$ for silicon and $3.9\text{x}10^{-4}$ for germanium).

W_{go} *is* the gap energy at 0 K.

Then, equation (5) can be written as:

$$n_i = BT^{3/2}e^{-W_{go}(1-CT)/2kT} \tag{7}$$

$$n_i = Be^{(W_{go}C/2k)}T^{3/2}e^{-(W_{go}/2kT)} \tag{8}$$

But $e^{W_{go}C/2k}$ is a constant and so can be combined with B to give a new constant B'. Therefore,

$$n_i = B'T^{3/2}e^{-(W_{go}/2kT)} \tag{9}$$

The constant B' can be either derived theoretically or determined experimentally. Commonly accepted values of W_{go} and B' for silicon are 1.2 electron volts and $3.88\text{x}10^{22}$, respectively.

From equations (2), (3), (4) and (9), the forward diode current can be written as:

$$I = qAB'^2T^3e^{-(W_{go}/kT)}\left(\frac{L_p}{\tau_p N_d} + \frac{L_n}{\tau_n N_a}\right)(e^{vq/\eta kT} - 1) \tag{10}$$

Equation (10) describes the current of the diode under forward bias with $vq/\eta kT >> 1$ as a function of two variables: the temperature T and the applied voltage v. Therefore, we can model the diode in the forward bias as shown in Fig. (3). The model has two inputs: the temperature T and the applied voltage v and an output which is either the diode current I or the diode current density J, if we divided by the area A in equation (10).

Assuming the parameters values shown in table (1) for a silicon diode, examples of the corresponding characteristic curves at different temperatures are plotted as shown in Fig.4.

V. Construction of Neural Network Model

The architecture of the proposed neural network to implement the diode model of Fig. 3 is as shown in Fig. 5. It consists of three layers; the input layer, the hidden layer and the output layer. The input layer that is considered as a dummy layer has two inputs: the temperature T and the forward voltage v.

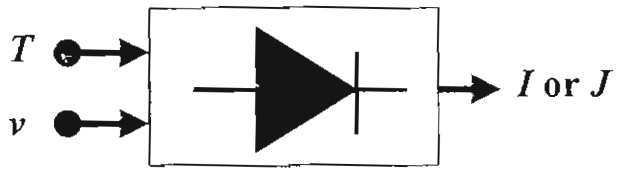

Fig. 3 The model of diode.

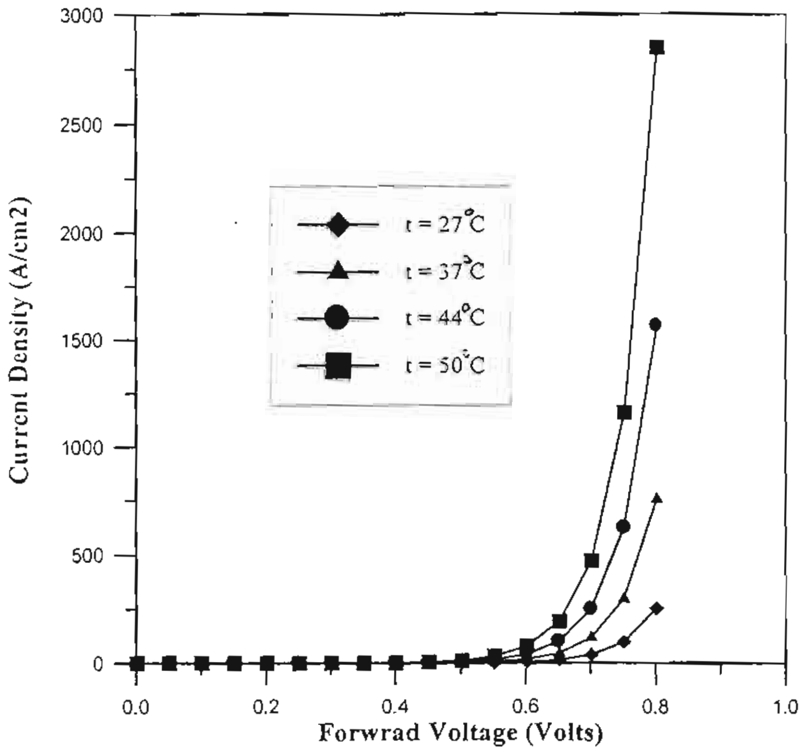

Fig. 4 V-I Diode Characteristic Curves at Different Temperatures.

Let $\mathbf{x}$ is the n-dimensional vector containing the parameters T and v, where N is the total number of input data samples. Then $\mathbf{x}$ can be written as:

$$x_i = [\,T_i \quad v_i\,]^T,\ i = 1, 2, 3, \ldots, N \tag{11}$$

The hidden layer is chosen to be consisting of 10 neurons, each has a sigmoid activating function. The output of each hidden neuron is:

$$z_h = f(\beta) = \frac{1}{1+e^{-\beta}} \qquad h = 1, 2, 3, \ldots, 10 \tag{12}$$

$$\text{and } \beta = \left(\sum_{i=1}^{N} x_i w_{ih}\right) + \mu_i \tag{13}$$

The output layer is one neuron and has a linear activating function. The output from the neural network is computed as:

$$y = \sum_{h=1}^{q} z_h v_h + \mu \tag{14}$$

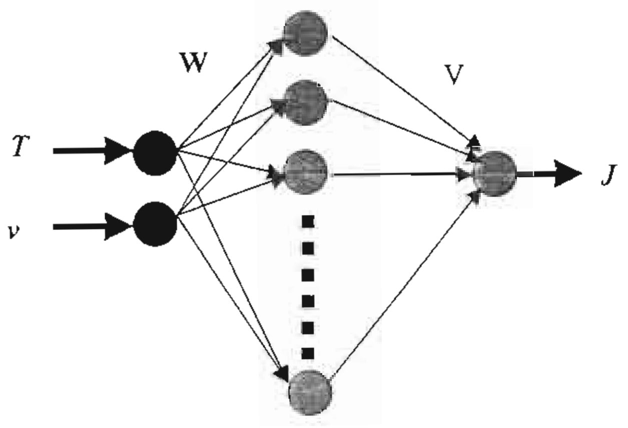

Fig. 5 Neural Network Model for Diode.

Table (1) Parameter values for the diode model.

Parameters	Values
η	2
B'	$3.88\text{x}10^{22}$
W_{go}	1.2 electron volts
L_p	$5\text{x}10^{-4}$ cm
L_n	$10\text{x}10^{-4}$ cm
τ_p	$2.8\text{x}10^{-8}$ second
τ_n	$5\text{x}10^{-8}$ second
N_d	$1\text{x}10^{16}$ cm^{-3}
N_a	$1\text{x}10^{18}$ cm^{-3}

VI. Learning Algorithms

The goal of learning is to adjust the weights values w_{ih}, v_h, and the threshold of each neuron in order to get the desired output when an input is presented to the network. The neural network model shown in Fig. 5 is trained using: the back propagation technique [1] and the genetic algorithm as follows:

a. Backpropagation Algorithm

The output function is given by:

$$output = \left[\sum_{j}^{10} \left(v_{jo} \sum_{i}^{10} f(w_{1i} v + w_{2i} T + b_i) \right) + b_o \right] \tag{15}$$

where

$f(.)$ is the sigmoid activation function,

v_{jo} is the weights between the hidden layer and the output layer,

w_{1i} is the weights between the first input and the hidden layer,

w_{2i} is the weights between the second input and the hidden layer,

b_i the bias of the hidden layer, and

b_o the bias of the output layer.

The rules of updating the weights are:

$$V_{jo}(new) = V_{jo}(old) + \eta \delta_o y_o \tag{16}$$

Where

V_{jo} is the weights from the hidden neuron "j "to the output neuron "o",

y_o is the actual output of the neuron "o",

η is the learning rate, and

δ is an error term for neuron "o" given by

$$\delta_o = d_o - y_o \tag{17}$$

where d_o is the desired (target) output of the neuron "o".

For the hidden neurons the rules are:

$$W_{ij}(new) = W_{ij}(old) + \eta \delta_j x_j \tag{18}$$

where

W_{ij} is the weights from the input neuron "i "to the hidden neuron "j",

x_j is the input, and

$$\delta_j = x_j(1 - x_j) \sum_k \delta_k w_{jk} \tag{19}$$

where k is over all neurons in the layer above neuron "j". The learning rate η is adaptive according to the following conditions:

$$\eta_{new} = \begin{cases} \eta_{old} * 1.1 & \text{if the error decreased} \\ \eta_{old} * 0.5 & \text{if the error increased} \end{cases} \tag{20}$$

Internal neuron thresholds are adapted in a similar manner by assuming they are connection weights on links from auxiliary constant-valued inputs.

b. Genetics Algorithm

The objective function (fitness function) used to train the neural network shown in Fig. 4 is written as:

$$fitness = \frac{1}{\sum_{i=1}^{N} (desired\_output - actual\_output)^2} \tag{21}$$

Where N is the total number of learning points.

In order to avoid the problems of assigning the starting generation of the required solution for the genetics algorithm. the initial values of the weights W, V and the thresholds are taken from the backpropagation after a suitable number of iterations. This approach not only speeds up the learning rate but also gives more accurate results too. The values obtained from the backpropagation are consider the starting generation in the genetics algorithm from which the remainder strings in the population can be generated according to the following equations:

$$\text{Upper Limit of } G_i = G_i + C \tag{22}$$

$$\text{Lower Limit of } G_i = G_i - C \tag{23}$$

Where G_i represents the value of any element of W, V or thresholds and C is an arbitrary constant. In our simulation C is taken to be equal to one. The genetic algorithm is summarized in the following steps:

step (1) Initialize the first population with the upper and lower boundaries described above.

step (2) Check the fitness value, if it is equal to the desired value then quit and if not continue.

step (3) Check the maximum number of generations, if it is equal to the required number then quit and if not continue.

step (4) Apply the genetic operators: reproduction, crossover and mutation to get the second generation and then go to step (2).

The parameter values shown in table (2) are used in our simulation.

Table (2) Parameter values for the diode model.

Parameter	**Value**
Crossover Probability	0.75
Mutation Probability	0.033
Population Size	30 strings (selected from 300 trials)
String Length	30 bits
Maximum Number of Generations	100

VII. Simulations and Results

The technique described above is applied for modeling PN junction diode operating at different working temperatures.

The backpropagtion algorithm is applied to learn the neural network mode of Fig. 3 with the diode forward characteristics described by equation (10). After 200.000 iterations the convergence becomes very slow and the learning time increases. Then. the GA is used to continue the learning process. The weights and biasing values obtained from the backpropagation algorithm after 200,000 iterations are used as the starting generation of the GA.

a. Backpropagation Algorithm Results

We begin to learn the neural network model of the diode using backprobagation with starting learning rate of 0.01. The output root mean square error (rms) has been reached the value of 0.099 after 500,000 iterations. The values of the weights and biasing obtained in this case are shown in table (3).

The biasing for output layer is 0.83301739648347 and the weights for the output layer is shown in table (4).

The model has been tested for both the trained data and untrained data for the same temperature. The results are in good agreement with the theoretical one. Also, untrained curves of different temperatures were tested and the results were as expected from the theoretical one. The model could cover a range of temperature from -50° C to 150° C.

b. Genetics Algorithm results

To speed up the learning proeess and to reduce the rms error as well, the genetic algorithm is applied to continue learning the diode model. The weights and biasing values of backpropagation after 200,000 iterations were used as seed values for the genetics algorithm. The genetics algorithm has reached to a root mean square error equals 0.005308 after 20 generations as shown in Fig. 6. The weights of hidden layer and its biasing are shown in table (5). The biasing of the output layer is 0.83992186514705 and the weights of it are shown in table (6).

Neural network has been learned at different temperatures, $T_1 = -50^{\circ}$ C, $T_2 = -25^{\circ}$ C, $T_3 = 27^{\circ}$C, $T_4 = 37^{\circ}$C, $T_5 = 44^{\circ}$C, $T_6 = 50^{\circ}$ C, $T_7 = 75^{\circ}$C, $T_8 = 100^{\circ}$ C, $T_9 = 125^{\circ}$ C, and $T_{10} = 150^{\circ}$ C. While the voltage v is swapped from 0.001 to 0.801 volts in step of 0.05 volts for each curve. After learning, the NN model was tested with the trained data and sample of the results at 125, 100. 75 $^{\circ}$ C are shown in Fig. 7.

Second, the NN model for diode has been tested using untrained data for the same temperatures above but at different forward voltage. In this case the voltage v swapped from v=0.0250 to 0.7750 volts with a step of 0.05 volts. The results at 50, 44 and 37 $^{\circ}$ C are shown in Fig. 8.

Finally, the NN model for diode has been tested with untrained curves at different temperature and the voltage v is swapped from 0.001 to 0.801 volts in step of 0.05 volts for each curve. The temperatures are $T_1 = -40^{\circ}$ C, $T_2 = -10^{\circ}$ C, $T_3 = 0^{\circ}$ C. $T_4 = 10^{\circ}$ C, $T_5 = 40^{\circ}$ C, $T_6 = 60^{\circ}$ C, $T_7 = 90^{\circ}$ C, $T_8 = 115^{\circ}$ C, $T_9 = 135^{\circ}$ C, and $T_{10} = 145^{\circ}$ C. The results at –40, -10 and 0 $^{\circ}$ C are as shown in Fig. 9.

Table (3) Weight Values for the Neural Network diode model using BP.

Biasing of the hidden layer	Weights for the first input to the hidden layer	Weights for the second input to the hidden layer
-3.63203958796369	0.01129143373524	2.48464588431426
3.03434822591552	-1.74016858725607	-1.62950881331518
2.57415202415192	4.51368374026143	2.82299437057592
-0.03508664457291	-3.24052597659191	-1.70308548920785
0.09169573495486	-2.05885757449650	-2.00063882303768
-0.08652723634296	-1.14965706537654	-2.99768757176890
-1.24587739784815	1.43990051877926	2.11441649558870
-10.4064628118841	6.51903746386485	-0.00551492994265
7.56366493726087	-3.97921849147037	0.51079041965370
-2.19588543454143	0.02020610357886	-48.7100712594018

Table (4)

The weights of the output layer
8.79254858465196
-5.08324358343775
6.90413529583811
-2.69331615646488
-2.95476559512257
-3.09833291784620
5.73201862167595
2.20878769295014
7.98646754508140
-17.1359362732924l

Table (5) Weight Values for the Neural Network diode model using GA.

Biasing of the hidden layer	Weights for the first input to the hidden layer	Weights for the second input to the hidden layer
-3.27976943642120	-0.11830200334727	2.36958819578845
3.16510949095806	-1.74893914352499	-1.42836890890812
2.75715008960999	4.42324713886997	2.48534024262596
-0.05251866887151	-2.73637946522270	-1.16704777172241
-0.24253904542544	-2.60894867753341	-2.14662997645995
-0.15789416540342	-1.14313292579502	-3.43286454494159
-1.42896138887851	1.48816427337071	2.16748327618607
-10.1903781292900	6.25126178262564	0.08776910336058
7.28824463967070	-3.74946830036166	0.43072171889100
-2.20556099876976	0.01840142614755	-49.2733283483977

Table (6)

The weights of the output layer
8.66062755878097
-5.37248540616668
7.13865084473732
-2.93492481357666
-3.20901147176624
-3.20588815244722
5.95450089316904
2.09669954949559
7.98207669215428
-17.239159368276

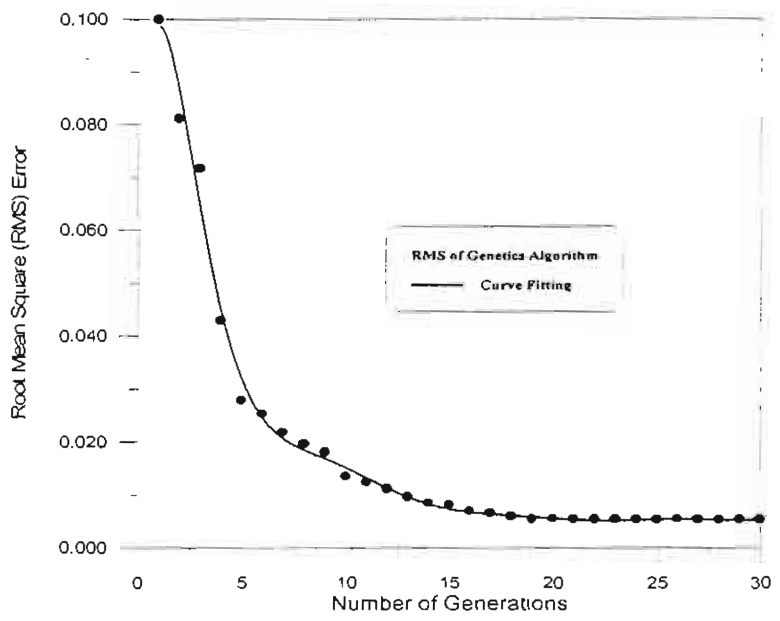

Fig. 6 The RMS of the output versus the number of generations.

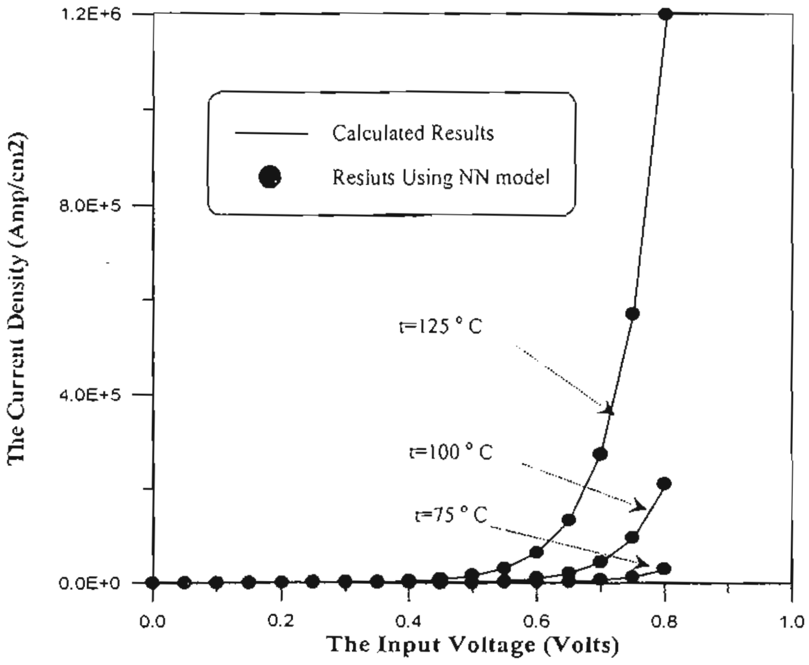

Fig. 7 The output of the NN model after training (trained date).

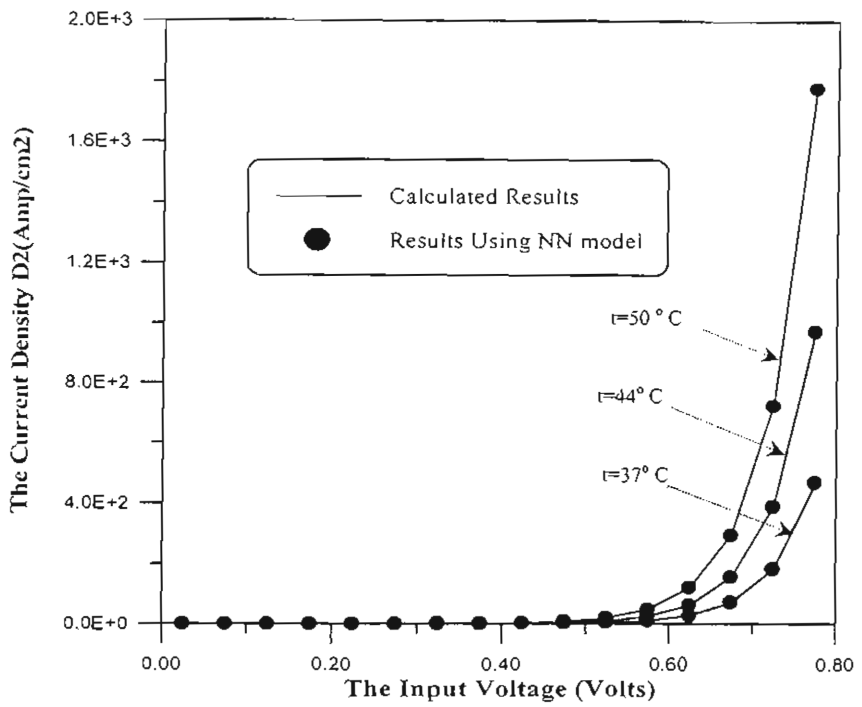

Fig. 8 The output of the NN model after training (untrained date).

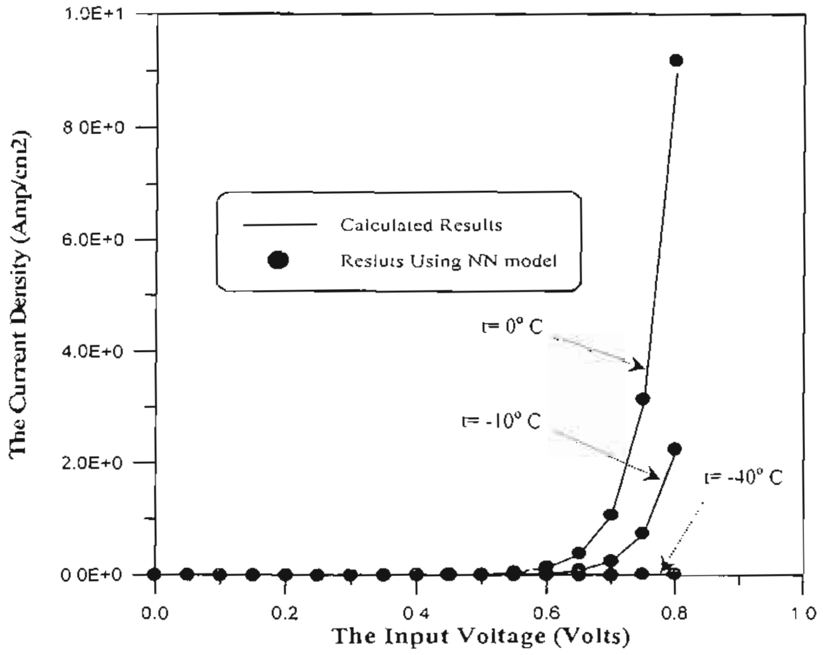

Fig. 9 The output of the NN model after training (untrained curves).

VIII. Conclusions

In this paper we have presented a novel technique to mode PN junction diodes. This technique is based on neural network and genetics algorithm. The obtained results have shown the efficiency and flexibility of the technique. By using neural network we have demonstrated the availability of modeling PN junction diode in the forward bias at different working temperature and by introducing genetic algorithm the learning speed as well as the accuracy have been highly improved. Even though a neural network model has no embedded electrical or physics equations, the model able to relate its output to the input parameters whether it has been trained with these parameters or not. The technique seems to be a promising one in the area of circuit analysis.

References

[1] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech, Signal Process. Mag.*, pp. 2-22, 1987.
[2] E. Sanchez-Sinencio and C. Lau, Eds., *Artificial Neural Networks*. New York: IEEE press, 1992.
[3] C. Mead, *Analog VLSI Neural System*, Addison-Wesely Publishing Company, 1989.
[4] M. Zurada, *Introduction to Artificial Neural System*, West Publishing Company, 1991.
[5] 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., June 1995.
[6] D. Rumelhart, G. Hinton and R. Williams, "Learning representation by backpropagation errors," *Nature*, vol. 323, pp. 533-536, 1986.
[7] W. S. McCulloch and W. Pitts, "A logical calculus of the idea immanent in nervous activity, " Bulletin of Mat. Bio., 5, pp. 115-133, 1943.
[8] R. Hecht-Nilson, *Neurocomputing*, Addison-Wesley Publishing Company, 1990.
[9] D. E. Goldberg, *Genetic Algorithms in Search. Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., 1989.
[10] E. S. Yang, *Microelectronic Devices*, MacGraw-Hill, Inc., 1988.
[11] C. L. Alley and K. W. Atwood, *Electronic Engineering*, John Wiley & Sons, New York, 1973.