# Double Gate MOSFET Modeling Based on Adaptive Neuro-Fuzzy Inference System for Nanoscale Circuit Simulation

Mohsen Hayati, Majid Seifi, and Abbas Rezaei

**As the conventional silicon metal-oxide-semiconductor field-effect transistor (MOSFET) approaches its scaling limits, quantum mechanical effects are expected to become more and more important. Accurate quantum transport simulators are required to explore the essential device physics as a design aid. However, because of the complexity of the analysis, it has been necessary to simulate the quantum mechanical model with high speed and accuracy. In this paper, the modeling of double gate MOSFET based on an adaptive neuro-fuzzy inference system (ANFIS) is presented. The ANFIS model reduces the computational time while keeping the accuracy of physics-based models, like non-equilibrium Green's function formalism. Finally, we import the ANFIS model into the circuit simulator software as a subcircuit. The results show that the compact model based on ANFIS is an efficient tool for the simulation of nanoscale circuits.**

**Keywords: Double gate MOSFET, adaptive neuro-fuzzy inference system, non-equilibrium Green's function, nanoscale circuits.**

Manuscript received Dec. 6, 2009; revised Apr. 20, 2010; accepted May 6, 2010.

Mohsen Hayati (Phone: +98 9188312041, email: mohsen_hayati@yahoo.com), Majid Seifi (email: majidsfy@gmail.com), and Abbas Rezaei (email: arezaei818@yahoo.com) are with the Electrical Engineering Department, Faculty of Engineering, Razi University, Kermanshah, Iran.

doi:10.4218/etrij.10.0109.0707

## I. Introduction

In the past decade, the dimensions of a metal-oxide-semiconductor field-effect transistor (MOSFET) have been scaled into the sub-50 nm regime [1], [2]. The double gate silicon-on-insulator structure [3], [4] has been proposed as a promising candidate to replace the conventional device structure. Having two gates ensures that no part of a channel is far away from the gate. This gives better control of the channel by the gate electrode. In addition, the voltage applied to the gate terminal determines the amount of current flowing through the channel. The schematic of symmetric double gate MOSFET (DG MOSFET) is shown in Fig. 1.

Although the operation of DG MOSFET is similar to the conventional MOSFET, the physics of this type of MOSFET is more complicated. Moreover, physical phenomena, such as the quantum mechanical effects, have to be considered. Therefore, simulation tools which can be applied to design nanoscale transistors in the future require new theories and modeling techniques that accurately and efficiently capture the physics of the quantum transport [5], [6]. Analytical models for such devices are required to be utilized in circuit simulators and circuit design tools. In order to achieve the required accuracy, several authors have considered various strategies for modeling the DG MOSFET [7]-[12].

The idea of modeling a DG MOSFET based on computational intelligence, that is, an artificial neural network (ANN) and adaptive neuro-fuzzy inference system (ANFIS), is given in [13] which concentrated on an ANN model implementation into circuit simulator software. Obtaining an

 © 2010 ETRI 

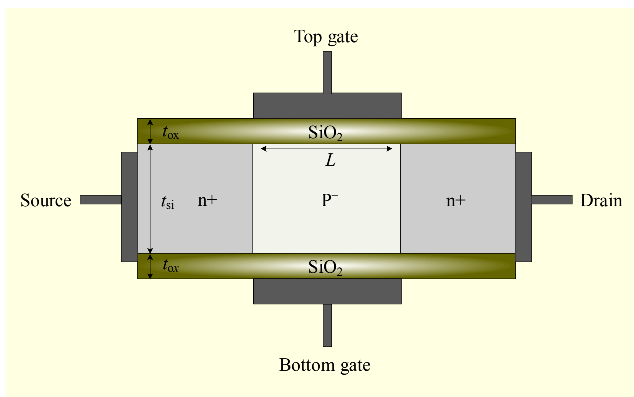


Fig. 1. Cross-section view of an *n*-channel DG MOSFET.

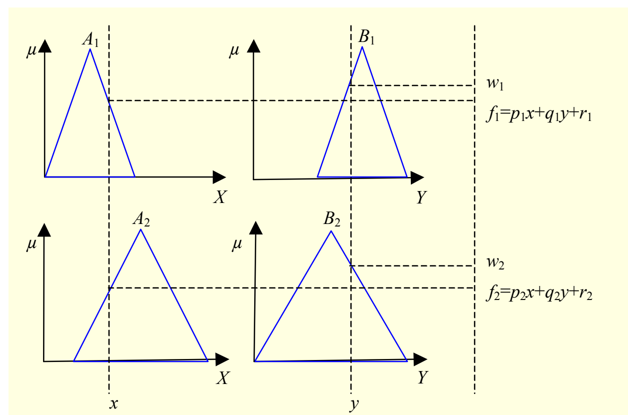


Fig. 2. Inference method of Sugeno model.

accurate model by an ANFIS and implementing the ANFIS model into circuit simulator software such as HSPICE have remained as challenges.

In order to simulate the nanoscale MOSFETs, the self consistent solution of the Poisson and Schrödinger equations can be performed within the non-equilibrium Green's function (NEGF) formalism. The NEGF formalism provides a powerful conceptual and computational framework for treating the quantum transport in nanodevices, and it is widely used as a method for nanoscale device simulation [14], [15].

In this paper, we have simulated the DG MOSFET within the NEGF formalism by using NanoMos 2.0 [16]. This software, developed by a group at Purdue University, is an accurate 2D simulator for a thin body, fully depleted, double gate *n*-type MOSFET. However, because of the complexity and time consuming nature of the quantum calculations for implementation in SPICE-type circuit simulators, it has been necessary to simulate the quantum mechanical model with high speed and accuracy.

This paper presents the applicability of an ANFIS for the simulation of the nanoscale circuits. The ANFIS could be used to provide a general link from measurements or device simulations to circuit simulation. The data thus obtained can be used as the target data set for optimizing the ANFIS architecture.

In this study, we have created the database for optimizing the ANFIS structure based on data obtained by simulation of DG MOSFET within the NEGF formalism. After optimization, we compared the simulation results of the proposed ANFIS model with NEGF simulation. Finally, the ANFIS model of the DG MOSFET was used as a subcircuit in the HSPICE software for the modeling and simulation of the nanoscale circuits.

## II. ANFIS

ANFIS is an adaptive network which permits the application of neural network topology together with fuzzy logic. The ANN provides effective learning methods and fast computations, whereas the fuzzy set theory allows thinking and reasoning capability for the fuzzy logic. The learning usually applies to the membership function parameters of the IF-THEN part of the fuzzy rules [17], [18].

At the computational level, the ANFIS can be regarded as a flexible mathematical structure that can approximate a large class of complex nonlinear systems to the desired degree of accuracy. In the fuzzy section, only the zeroth-order or first-order Sugeno inference system or the Tsukamoto inference system can be used [19]. For simplicity, we assume that the fuzzy inference system has two inputs ($x$, $y$) and one output ($f$). For a first-order Sugeno fuzzy model, a typical rule set with fuzzy-based IF-THEN rules can be expressed as follows:

Rule 1: If $x$ is $A_1$ and $y$ is $B_1$, then

$$f_1 = p_1 x + q_1 y + r_1. \quad (1)$$

Rule 2: If $x$ is $A_2$ and $y$ is $B_2$, then

$$f_2 = p_2 x + q_2 y + r_2. \quad (2)$$

Here $p_i$, $q_i$, and $r_i$ are linear output parameters (consequent parameters), where $i$ = 1, 2. The related inference method is shown in Fig. 2.

One possible ANFIS architecture to implement these two rules is shown in Fig. 3. It should be noted that the circle indicates a fixed node, whereas the square indicates an adaptive node.

A description of the layers in the network is summarized as follows:

Layer 1: Every node in this layer is an adaptive node with a node function.

$$O_{1,i} = \mu_{Ai}(x), \qquad i = 1, 2, \cdots, \quad (3)$$

$$O_{1,i} = \mu_{Bi\text{-}2}(y), \qquad i = 3, 4, \cdots, \quad (4)$$

where $i$ is the membership grade of a fuzzy set ($A_1$, $A_2$, $B_1$, $B_2$) and $O_{1,i}$ is the output of the node $i$ in Layer 1. The membership

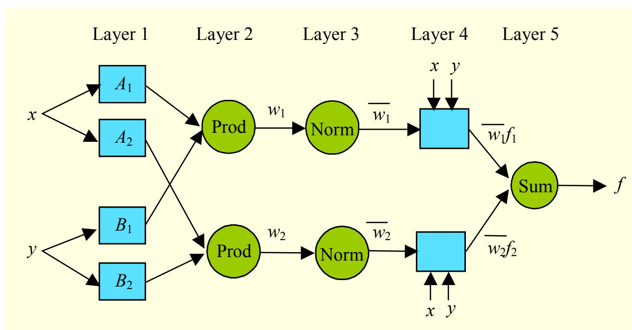


Fig. 3. ANFIS architecture based on Sugeno.

Table 1. Data ranges used for building the proposed ANFIS model.

| Range | $L$ (nm) | $V_{ds}$ (V) | $V_{gs}$ (V) | $t_{si}$ (nm) | $t_{ox}$ (nm) |
|---|---|---|---|---|---|
| Min | 5 | 0 | 0 | 3 | 1 |
| Max | 50 | 0.5 | 0.5 | 5 | 2 |

function that has been used in this study is the Gaussian function given by

$$\mu_A(x) = \exp\left(\frac{-0.5(x-c)^2}{\sigma^2}\right), \tag{5}$$

where $c$ and $\sigma$ are referred to as premise parameters.

Layer 2: Each node in this layer is a fixed node and calculates the firing strength of a rule via multiplication. The outputs are given by

$$O_{2,i} = w_i = \mu_{Ai}(x).\,\mu_{Bi}(y), \qquad i = 1, 2. \tag{6}$$

In general, any other T-norm operator performing fuzzy AND method can be used as the node function in this layer.

Layer 3: Every node in this layer is also fixed and performs a normalization of the firing strength from the previous layer. The outputs of this layer are called normalized firing strengths and are given by

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2}, \qquad i = 1, 2, \tag{7}$$

where $O_{3,i}$ denotes the Layer 3 output.

Layer 4: In this layer, all nodes are adaptive, and the output of a node is the product of the normalized firing strength and a first-order polynomial given by

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i), \quad i = 1, 2, \tag{8}$$

where $\overline{w_i}$ is the output of Layer 3, and ($p_i$, $q_i$, $r_i$) is the parameter set. Parameters in this layer are referred to as the consequent parameters.

Layer 5: The single node in this layer is a fixed node and computes the overall output as the summation of contribution from each rule:

$$O_{5,1} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}, \qquad i = 1, 2, \tag{9}$$

where $O_{5,1}$ denotes the Layer 5 output.

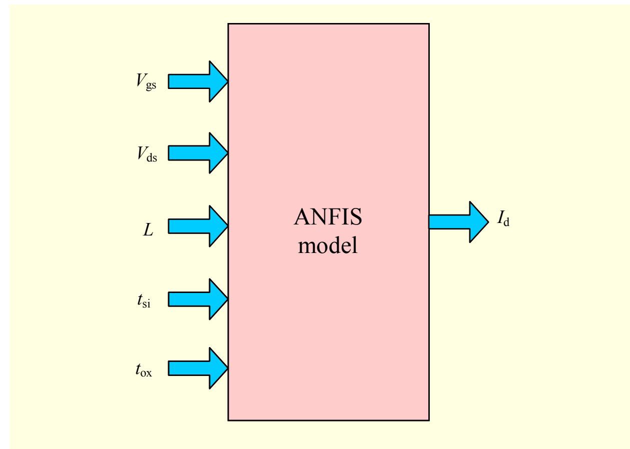


Fig. 4. Simplified overview of ANFIS model.

A hybrid learning algorithm [20] is used for ANFIS training, and it consists of two stages: forward pass and backward pass. In the forward pass, the consequent parameters are identified by the least squares estimation, and in the backward pass, the premise parameters are updated by the gradient descent. Further details about ANFIS and these learning algorithms can be found in [21].

In this paper, we have built the ANFIS model to relate input parameters ($L$, $V_{ds}$, $V_{gs}$, $t_{ox}$, and $t_{si}$) to output parameter $I_d$, where $L$ is the channel length, $V_{ds}$ is the drain source voltage, $V_{gs}$ is the gate source voltage, $t_{si}$ is the silicon film thickness, $t_{ox}$ is the gate oxide thickness, and $I_d$ is the drain current in DG MOSFET.

The minimum and maximum data ranges used to develop the ANFIS model are shown in Table 1.

A simplified overview of the proposed ANFIS model is shown in Fig. 4.

## III. Result and Discussion

### 1. Modeling Results

In order to optimize the ANFIS model, about 700 data were obtained by simulation of the DG MOSFET by using NanoMos 2.0. About 70% of the data was selected to train the ANFIS model, and the remaining was used to test the performance of the trained ANFIS model. Given that the testing and training samples must be different, they were selected randomly from the original database (NEGF

Table 2. Optimal architecture and specification of the proposed ANFIS model.

| Type | Sugeno |
|---|---|
| Inputs/outputs | 5/1 |
| No. of input membership functions | 24 for each input |
| No. of output membership functions | 24 |
| Input membership function type | Gaussian |
| Output membership function type | Linear |
| No. of fuzzy rules | 24 |
| No. of linear parameters | 144 |
| No. of nonlinear parameters | 480 |
| No. of epochs | 1,000 |

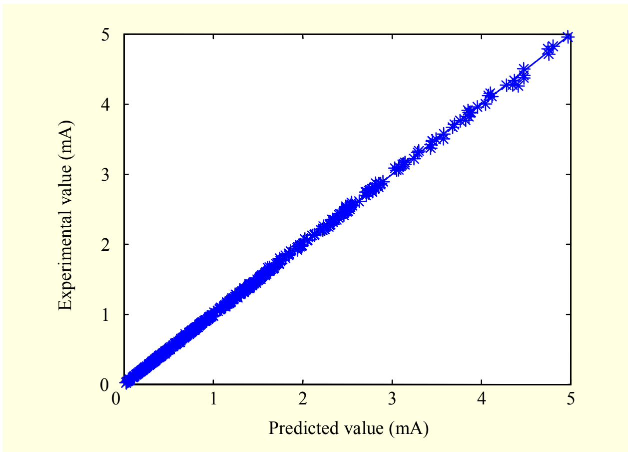


Fig. 5. Comparison of the numerical (NEGF) and predicted (ANFIS) results for training data.

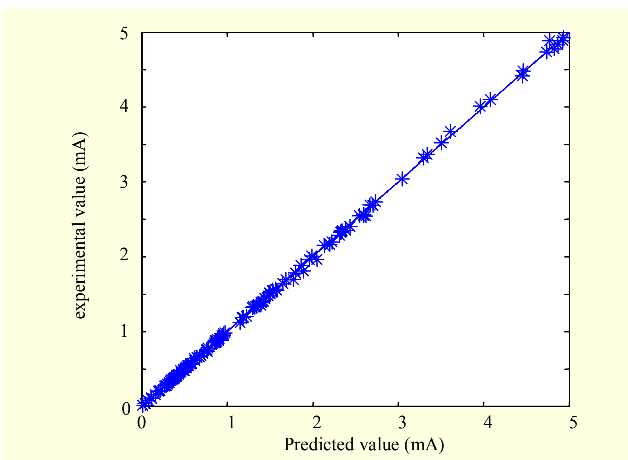


Fig. 6. Comparison of the numerical (NEGF) and predicted (ANFIS) results for testing data.

simulation). The final ANFIS architecture used in this study is shown in Table 2. The improved optimized ANFIS model structure has 24 membership functions in comparison with the 42 membership functions in [13]. This optimized model has made it possible to implement the ANFIS model into the circuit simulator software.

Table 3. Relative errors for training and testing results of the proposed ANFIS model.

| RE% | Train | Test |
|---|---|---|
| Min | 0.0002 | 0.022 |
| Max | 6.01 | 9.48 |
| Mean | 0.99 | 2.3 |

The training and testing results of the proposed ANFIS model are shown in Figs. 5 and 6 and Table 3, where the mean relative error (MRE) is given by

$$\mathrm{MRE} = \frac{1}{N} \times \sum_{i=1}^{N} \left| \frac{X_i(Num) - X_i(Pred)}{X_i(Num)} \right|, \tag{10}$$

where $X(Num)$ and $X(Pred)$ stand for numerical (NEGF simulation) and predicted (ANFIS) values, respectively, and $N$ is the number of data.

As shown in Figs. 5 and 6 and Table 3, for the best ANFIS configuration obtained in this study, almost 100% of the submitted cases were learnt correctly, and the MRE% of the ANFIS output ($I_d$) was less than 2%.

The comparison of current-voltage characteristics ($I_d$-$V_{ds}$ and $I_d$-$V_{gs}$) between the NEGF simulation and the proposed ANFIS model for a symmetric DG MOSFET with $L$=10 nm, $t_{si}$=3 nm, and $t_{ox}$=2 nm are shown in Figs. 7 and 8. The comparison of transconductance, $g_m = \frac{\partial I_d}{\partial V_{gs}}$, between NanoMos 2.0 and the ANFIS is shown in Fig. 9.

The results show that there is an excellent agreement between the numerical and predicted values with the least errors. Also, we can extend the number of input parameters of the ANFIS model to other parameters like temperature and gate direct tunneling current.

Besides high accuracy, the calculations speed of the proposed ANFIS model to obtain the results is extremely high. On the same workstation (1.8 GHz), the simulation time for obtaining drain current ($I_d$) at several bias points with different $V_{gs}$ and $V_{ds}$ (Figs. 7 and 8) was measured for the ANFIS model and NanoMos 2.0. We obtained the central processing unit (CPU) time requirements for the ANFIS and NanoMos 2.0 in the MATLAB environment by using CLOCK command at the beginning and the end of the MATLAB program (M-file). The difference between the times obtained by CLOCK commands shows the CPU time requirement for each simulation.

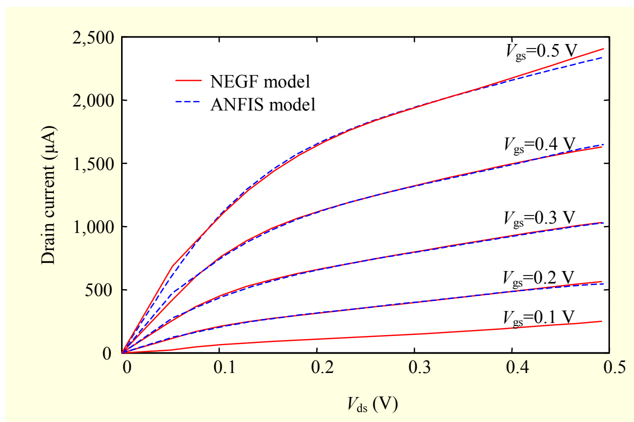


Fig. 7. Current-voltage characteristics curve ($I_d$-$V_{ds}$) for DG MOSFET ($L$=10 nm, $t_{si}$=3 nm and $t_{ox}$=2 nm) using NEGF simulation (solid line) and ANFIS model (dashed line).

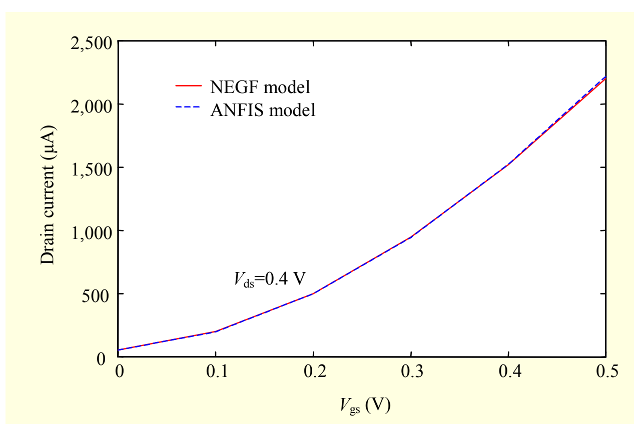


Fig. 8. Current-voltage characteristics curve ($I_d$-$V_{gs}$) for DG MOSFET ($L$=10 nm, $t_{si}$=3 nm and $t_{ox}$=2 nm) using NEGF simulation (solid line) and ANFIS model (dashed line).

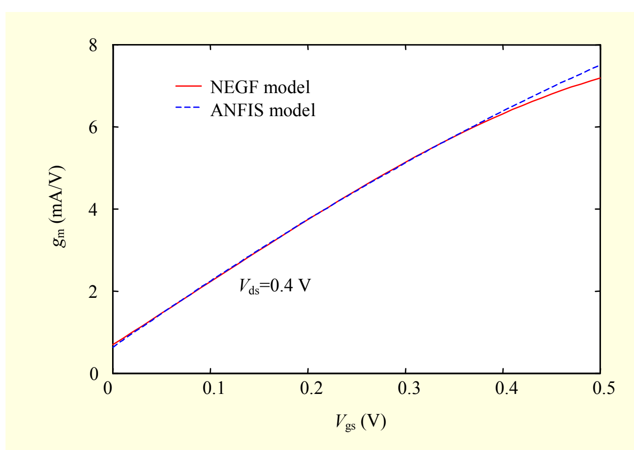


Fig. 9. Transconductance curve for DG MOSFET ($L$=10 nm, $t_{si}$=3 nm and $t_{ox}$=2 nm) using NEGF simulation (solid line) and ANFIS model (dashed line).

Table 4. Average CPU time comparison between ANFIS model and NanoMos 2.0.

| No. of simulation points | Time (s) | |
|---|---|---|
| | NanoMOS 2.0 | ANFIS |
| 5 | 475 | Less than 0.01 |
| 20 | 3,470 | 0.4 |
| 50 | 10,950 | 1.3 |
| 100 | 20,250 | 2 |

The average CPU time comparison between the ANFIS model and NanoMos 2.0 (NEGF simulation) is shown in Table 4. It is clear that the ANFIS model is much faster than NanoMos 2.0.

In this paper, we have proposed the ANFIS model as an improved approach over the ANN model in [5]. The ANFIS model could significantly reduce the output errors to less than 2% in comparison with the ANN model with output errors of less than 5%. Another advantage of the ANFIS model in comparison with the ANN model is the lower number of epochs needed to reach convergence (1,000 in comparison with 10,000). Therefore, the training time for the ANFIS model is definitely less than the required time for designing similar models using pure neural networks. It means that the ANFIS model is better than ANN for redeveloping the model and increasing the input parameters. It may be noted that for an ANN model, we have to perform a trial and error process to develop the optimal network architecture, while the ANFIS model does not require such a procedure. This is because the ANFIS is more transparent, and it is possible to obtain input-output relationships from membership functions and IF-THEN rules.

## 2. Implementation into HSPICE

Developing a new equivalent circuit model is often time consuming, and it is difficult to create a formula for non-linear elements when a physics of a new device is very complicated. With the high computational speed and good accuracy of the ANFIS model in the simulation of the DG MOSFET, we can use the ANFIS as a neuro-fuzzy behavioral model in a circuit simulator after translating the ANFIS equations into an appropriate syntax.

The main element contributing to the nonlinear behavior of the DG MOSFET device is the drain current $I_d$ that is the function of the $L$, $V_{ds}$, $V_{gs}$, $t_{ox}$, and $t_{si}$ parameters. Therefore, in HSPICE implementation, the MOSFET may be considered as a voltage-controlled current source [22]. Its general syntax is

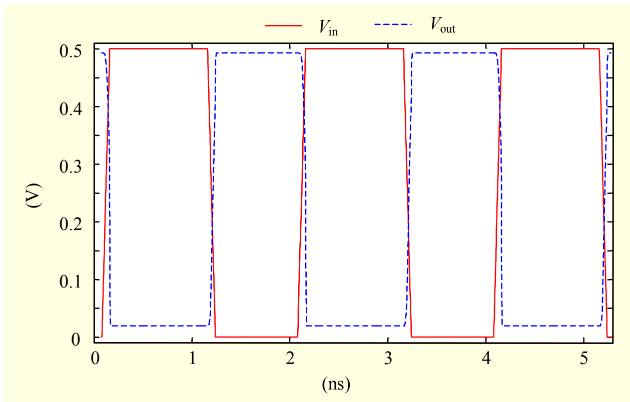


Fig. 10. HSPICE input and output signals of the ANFIS inverter.

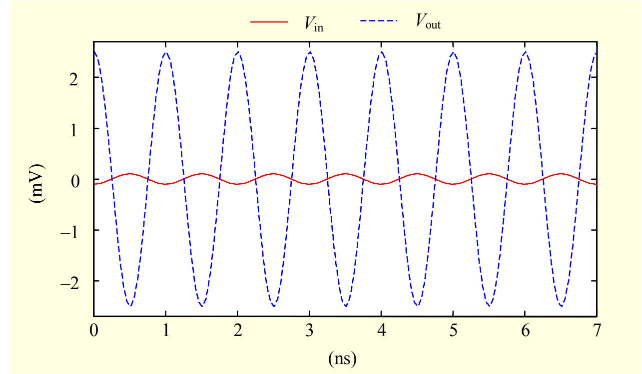


Fig. 12. HSPICE input and output signals of the ANFIS common-source amplifier.

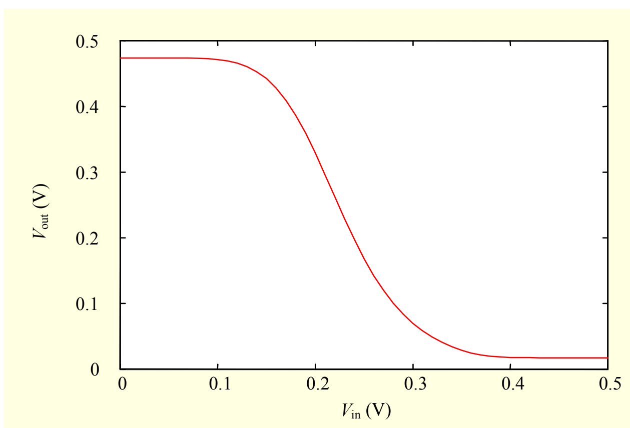


Fig. 11. Predicted transfer curve using the ANFIS model for the DG MOSFET inverter.

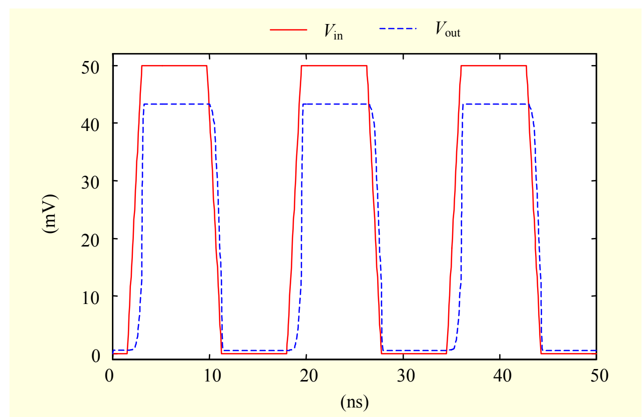


Fig. 13. HSPICE input and output signals of the ANFIS common-drain amplifier.

given as follows:

*gname node1 node2* [*cur*='mathematical expression'].

This statement creates a current source according to the expression given between quotes [23]. In our case, this expression is determined by the ANFIS equations. The gate current of the MOS transistor is always neglected and could be modeled by a null current source. Therefore, in HSPICE simulations, the implemented DG MOSFET model is a subcircuit constituted of a null current source between gate and source and a voltage-controlled current source between drain and source where its expression is given by the ANFIS equations.

After optimizing the proposed ANFIS structure, all parameters of Gaussian input membership functions and linear output parameters are obtained. The firing strength of each rule can be calculated via multiplication of the Gaussian input membership functions (fuzzy AND), and the final output of the model (drain current) is calculated as the summation of the contribution from each rule. We implemented these equations in HSPICE to calculate the drain current of the DG MOSFET. The implementation approach takes advantage of the subcircuit definition option in HSPICE. The general form is:

.SUBCKT DGmos $N_{drain}$ $N_{gate}$ $N_{source}$

Igate $N_{gate}$ $N_{source}$ 0

Gdrain $N_{drain}$ $N_{source}$ cur='Sugeno ANFIS equations to calculate the drain current'

.ENDS,

where $N_{drain}$ is the drain-node, $N_{source}$ is the source-node, $N_{gate}$ is the gate-node, Igate is a null current source between gate and source, Gdrain is the voltage-controlled current source ($I_d$), and DGmos is the device name. In order to validate the proposed ANFIS model, we simulated the DG MOSFET inverter constituted of a DG MOSFET with $L$=10 nm, $t_{si}$=3 nm, and $t_{ox}$=2 nm, in series with a resistor placed in drain and common-source amplifier. More details about the simulator can be found in the Appendix.

In order to establish the correctness of the ANFIS model, many examples were included. The HSPICE input/output signals and transfer curve of proposed ANFIS inverter gate are shown in Figs. 10 and 11. Also, the HSPICE input/output signals for common-source and common-drain amplifiers are shown in Figs. 12 and 13.

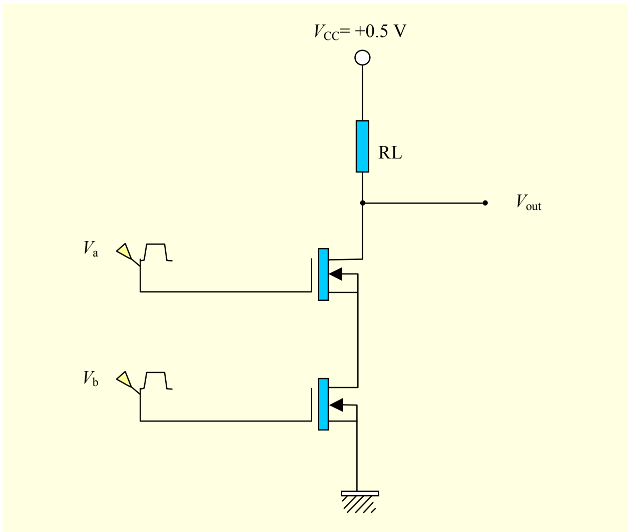


Fig. 14. Circuit diagram of the ANFIS NAND gate.

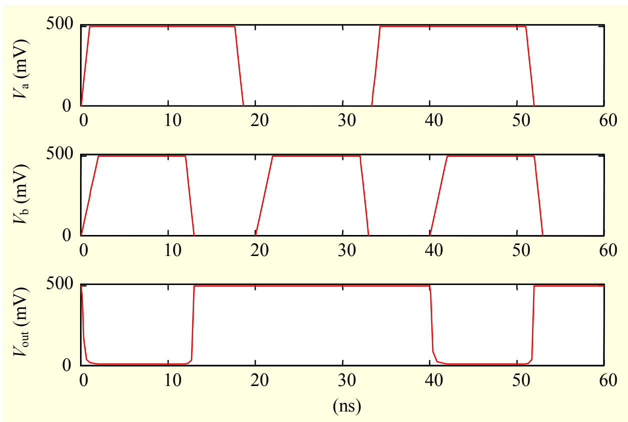


Fig. 15. HSPICE input and output signals of the ANFIS NAND gate.

The circuit diagram and input/output signals of DG MOSFET NAND gate can be found in Figs. 14 and 15. The CPU time for these simulations was about 0.4 s.

The required CPU time for simulating the ANFIS model in HSPICE is less than the ANN model [13], that is, 0.4 s in comparison with 0.9 s. It means that the ANFIS model is faster than the ANN model for simulation of circuits.

It is important to note that the proposed DG MOSFET model can be extended to other practical circuits with many devices. Therefore, this model can be used as the interface between device modeling and circuit simulators like HSPICE, PSPICE, and ADS in order to have a simple and accurate nanoscale circuits simulator.

## IV. Conclusion

In this paper, we have presented the applicability of an ANFIS approach to modeling and the simulation of a DG MOSFET. We have used 2D numerical NEGF simulation of the current-voltage characteristics of an undoped symmetric DG MOSFET by using NanoMos 2.0. With this numerical model, the required database has been created in order to optimize the ANFIS structure. The comparison between numerical (NEGF simulation) and predicted (ANFIS model) values has shown that there is an excellent agreement between them with the least errors. Moreover, we have measured the CPU time for the ANFIS model against the NEGF model. The result has shown that the ANFIS model is much faster than NanoMos 2.0 (NEGF simulation). Also, we have used the ANFIS as a neuro-fuzzy behavioral model in HSPICE after translating the ANFIS equations into an appropriate syntax. The simulation results by HSPICE have shown that the proposed DG MOSFET model is suitable for using as the interface between device modeling and a circuit simulator in order to improve the simulation speed and accuracy of the nanoscale devices.

## V. Appendix

The ANFIS Model Implementation into HSPICE

```
**Define the Gaussian input membership functions.
.param gaussmf(x, B, C)='exp(-pow((x-C),2)/(2*pow(B,2)))'
**Bi, Ci, (i=1,2,…,24) are the parameters of Gaussian input membership
   functions and Pi, Qi, Ri, Si, Ti, Ui, (i=1,2,…,24) are the linear output
   parameters (consequent parameters) that are obtained from optimized ANFIS
   structure.
**Wi (i=1,2,…,24) are the firing strength of each rule and Fi (i=1,2,…,24) are the
   Sugeno fuzzy model rules.
**n1, n2 and n3 stand for drain, gate and source nodes, respectively.
**The gate current (igate) is assumed to be zero.
**Define the input parameters (channel length, silicon film thickness and gate
oxide thickness).
.param Lg=10 tox=2 tsi=3

** The firing strength of each rule can be calculated via multiplication of the
   Gaussian input membership functions (fuzzy AND)
.param W1='gaussmf(par(drain), B1drain,C1drain)*gaussmf(par(gate),
   B1gate,C1gate)*gaussmf(par(tsi), B1tsi,C1tsi)* gaussmf(par(tox),
   B1tox,C1tox)* gaussmf(par(Lg), B1Lg,C1Lg)'
.param W2='gaussmf(par(drain), B2drain,C2drain)*gaussmf(par(gate),
   B2gate,C2gate)*gaussmf(par(tsi), B2tsi,C2tsi)* gussmf(par(tox),
   B2tox,C2tox)* gaussmf(par(Lg), B2Lg,C2Lg)'
.param W3='gaussmf(par(drain), B3drain,C3drain)*gaussmf(par(gate),
   B3gate,C3gate)*gaussmf(par(tsi), B3tsi,C3tsi)* gaussmf(par(tox),
   B3tox,C3tox)* gaussmf(par(Lg), B3Lg,C3Lg)'
.param W4='gaussmf(par(drain), B4drain,C4drain)*gaussmf(par(gate),
   B4gate,C4gate)*gaussmf(par(tsi), B4tsi,C4tsi)* gaussmf(par(tox),
   B4tox,C4tox)* gaussmf(par(Lg), B4Lg,C4Lg)'
.param W5='gaussmf(par(drain), B5drain,C5drain)*gaussmf(par(gate),
```

```
B5gate,C5gate)*gaussmf(par(tsi), B5tsi,C5tsi)* gaussmf(par(tox),
B5tox,C5tox)* gaussmf(par(Lg), B5Lg,C5Lg)'
.param W6='gaussmf(par(drain), B6drain,C6drain)*gaussmf(par(gate),
B6gate,C6gate)*gaussmf(par(tsi), B6tsi,C6tsi)* gaussmf(par(tox),
B6tox,C6tox) * gaussmf(par(Lg), B6Lg,C6Lg)'
.param W7='gaussmf(par(drain), B7drain,C7drain)*gaussmf(par(gate),
B7gate,C7gate)*gaussmf(par(tsi), B7tsi,C7tsi)* gaussmf(par(tox),
B7tox,C7tox)* gaussmf(par(Lg), B7Lg,C7Lg)'
.param W8='gaussmf(par(drain), B8drain,C8drain)*gaussmf(par(gate),
B8gate,C8gate)*gaussmf(par(tsi), B8tsi,C8tsi)* gaussmf(par(tox),
B8tox,C8tox) *    gaussmf(par(Lg), B8Lg,C8Lg)'
.param W9='gaussmf(par(drain), B9drain,C9drain)*gaussmf(par(gate),
B9gate,C9gate)*gaussmf(par(tsi), B9tsi,C9tsi)* gaussmf(par(tox),
B9tox,C9tox)* gaussmf(par(Lg), B9Lg,C9Lg)'
.param W10='gaussmf(par(drain), B10drain,C10drain)*gaussmf(par(gate),
B10gate,C10gate)*gaussmf(par(tsi), B10tsi,C10tsi)* gaussmf(par(tox),
B10tox,C10tox)* gaussmf(par(Lg), B10Lg,C10Lg)'
.param W11='gaussmf(par(drain), B11drain,C11drain)*gaussmf(par(gate),
B11gate,C11gate)*gaussmf(par(tsi), B11tsi,C11tsi)* gaussmf(par(tox),
B11tox,C11tox)* gaussmf(par(Lg), B11Lg,C11Lg)'
.param W12='gaussmf(par(drain), B12drain,C12drain)*gaussmf(par(gate),
B12gate,C12gate)*gaussmf(par(tsi), B12tsi,C12tsi)* gaussmf(par(tox),
B12tox,C12tox)* gaussmf(par(Lg), B12Lg,C12Lg)'
.param W13='gaussmf(par(drain), B13drain,C13drain)*gaussmf(par(gate),
B13gate,C13gate)*gaussmf(par(tsi), B13tsi,C13tsi)* gaussmf(par(tox),
B13tox,C13tox)* gaussmf(par(Lg), B13Lg,C13Lg)'
.param W14='gaussmf(par(drain), B14drain,C14drain)*gaussmf(par(gate),
B14gate,C14gate)*gaussmf(par(tsi), B14tsi,C14tsi)* gaussmf(par(tox),
B14tox,C14tox)* gaussmf(par(Lg), B14Lg,C14Lg)'
.param W15='gaussmf(par(drain), B15drain,C15drain)*gaussmf(par(gate),
B15gate,C15gate)*gaussmf(par(tsi), B15tsi,C15tsi)* gaussmf(par(tox),
B15tox,C15tox)* gaussmf(par(Lg), B15Lg,C15Lg)'
.param W16='gaussmf(par(drain), B16drain,C16drain)*gaussmf(par(gate),
B16gate,C16gate)*gaussmf(par(tsi), B16tsi,C16tsi)* gaussmf(par(tox),
B16tox,C16tox)* gaussmf(par(Lg), B16Lg,C16Lg)'
.param W17='gaussmf(par(drain), B17drain,C17drain)*gaussmf(par(gate),
B17gate,C17gate)*gaussmf(par(tsi), B17tsi,C17tsi)* gaussmf(par(tox),
B17tox,C17tox)* gaussmf(par(Lg), B17Lg,C17Lg)'
.param W18='gaussmf(par(drain), B18drain,C18drain)*gaussmf(par(gate),
B18gate,C18gate)*gaussmf(par(tsi), B18tsi,C18tsi)* gaussmf(par(tox),
B18tox,C18tox)* gaussmf(par(Lg), B18Lg,C18Lg)'
.param W19='gaussmf(par(drain), B19drain,C19drain)*gaussmf(par(gate),
B19gate,C19gate)*gaussmf(par(tsi), B19tsi,C19tsi)* gaussmf(par(tox),
B19tox,C19tox)* gaussmf(par(Lg), B19Lg,C19Lg)'
.param W20='gaussmf(par(drain), B20drain,C20drain)*gaussmf(par(gate),
B20gate,C20gate)*gaussmf(par(tsi), B20tsi,C20tsi)* gaussmf(par(tox),
B20tox,C20tox)* gaussmf(par(Lg), B20Lg,C20Lg)'
.param W21='gaussmf(par(drain), B21drain,C21drain)*gaussmf(par(gate),
B21gate,C21gate)*gaussmf(par(tsi), B21tsi,C21tsi)* gaussmf(par(tox),
B21tox,C21tox)* gaussmf(par(Lg), B21Lg,C21Lg)'
.param W22='gaussmf(par(drain), B22drain,C22drain)*gaussmf(par(gate),
B22gate,C22gate)*gaussmf(par(tsi), B22tsi,C22tsi)* gaussmf(par(tox),
B22tox,C22tox)* gaussmf(par(Lg), B22Lg,C22Lg)'
.param W23='gaussmf(par(drain), B23drain,C23drain)*gaussmf(par(gate),
B23gate,C23gate)*gaussmf(par(tsi), B23tsi,C23tsi)* gaussmf(par(tox),
B23tox,C23tox)* gaussmf(par(Lg), B23Lg,C23Lg)'
.param W24='gaussmf(par(drain), B24drain,C24drain)*gaussmf(par(gate),
B24gate,C24gate)*gaussmf(par(tsi), B24tsi,C24tsi)* gaussmf(par(tox),
B24tox,C24tox)* gaussmf(par(Lg), B24Lg,C24Lg)'

.param F1='(P1drain *par(drain)+ Q1gate *par(gate)+ R1tsi *par(tsi)+ S1tox
*par(tox)+ T1Lg *par(Lg)+ U1)'
.param F2='(P2drain *par(drain)+ Q2gate *par(gate)+ R2tsi *par(tsi)+ S2tox
*par(tox)+ T2Lg *par(Lg)+ U2)'
.param F3='(P3drain *par(drain)+ Q3gate *par(gate)+ R3tsi *par(tsi)+ S3tox
*par(tox)+ T3Lg *par(Lg)+ U3)'
.param F4='(P4drain *par(drain)+ Q4gate *par(gate)+ R4tsi *par(tsi)+ S4tox
*par(tox)+ T4Lg *par(Lg)+ U4)'
.param F5='(P5drain *par(drain)+ Q5gate *par(gate)+ R5tsi *par(tsi)+ S5tox
*par(tox)+ T5Lg *par(Lg)+ U5)'
.param F6='(P6drain *par(drain)+ Q6gate *par(gate)+ R6tsi *par(tsi)+ S6tox
*par(tox)+ T6Lg *par(Lg)+ U6)'
.param F7='(P7drain *par(drain)+ Q7gate *par(gate)+ R7tsi *par(tsi)+ S7tox
*par(tox)+ T7Lg *par(Lg)+ U7)'
.param F8='(P8drain *par(drain)+ Q8gate *par(gate)+ R8tsi *par(tsi)+ S8tox
*par(tox)+ T8Lg *par(Lg)+ U8)'
.param F9='(P9drain *par(drain)+ Q9gate *par(gate)+ R9tsi *par(tsi)+ S9tox
*par(tox)+ T9Lg *par(Lg)+ U9)'
.param F10='(P10drain *par(drain)+ Q10gate *par(gate)+ R10tsi *par(tsi)+
S10tox *par(tox)+ T10Lg *par(Lg)+ U10)'
.param F11='(P11drain *par(drain)+ Q11gate *par(gate)+ R11tsi *par(tsi)+
S11tox *par(tox)+ T11Lg *par(Lg)+ U11)'
.param F12='(P12drain *par(drain)+ Q12gate *par(gate)+ R12tsi *par(tsi)+
S12tox *par(tox)+ T12Lg *par(Lg)+ U12)'
.param F13='(P13drain *par(drain)+ Q13gate *par(gate)+ R13tsi *par(tsi)+
S13tox *par(tox)+ T13Lg *par(Lg)+ U13)'
.param F14='(P14drain *par(drain)+ Q14gate *par(gate)+ R14tsi *par(tsi)+
S14tox *par(tox)+ T14Lg *par(Lg)+ U14)'
.param F15='(P15drain *par(drain)+ Q15gate *par(gate)+ R15tsi *par(tsi)+
S15tox *par(tox)+ T15Lg *par(Lg)+ U15)'
.param F16='(P16drain *par(drain)+ Q16gate *par(gate)+ R16tsi *par(tsi)+
S16tox *par(tox)+ T16Lg *par(Lg)+ U16)'
.param F17='(P17drain *par(drain)+ Q17gate *par(gate)+ R17tsi *par(tsi)+
S17tox *par(tox)+ T17Lg *par(Lg)+ U17)'
.param F18='(P18drain *par(drain)+ Q18gate *par(gate)+ R18tsi *par(tsi)+
S18tox *par(tox)+ T18Lg *par(Lg)+ U18)'
.param F19='(P19drain *par(drain)+ Q19gate *par(gate)+ R19tsi *par(tsi)+
S19tox *par(tox)+ T19Lg *par(Lg)+ U19)'
.param F20='(P20drain *par(drain)+ Q20gate *par(gate)+ R20tsi *par(tsi)+
S20tox *par(tox)+ T20Lg *par(Lg)+ U20)'
.param F21='(P21drain *par(drain)+ Q21gate *par(gate)+ R21tsi *par(tsi)+
S21tox *par(tox)+ T21Lg *par(Lg)+ U21)'
.param F22='(P22drain *par(drain)+ Q22gate *par(gate)+ R22tsi *par(tsi)+
S22tox *par(tox)+ T22Lg *par(Lg)+ U22)'
.param F23='(P23drain *par(drain)+ Q23gate *par(gate)+ R23tsi *par(tsi)+
S23tox *par(tox)+ T23Lg *par(Lg)+ U23)'
.param F24='(P24drain *par(drain)+ Q24gate *par(gate)+ R24tsi *par(tsi)+
S24tox *par(tox)+ T24Lg *par(Lg)+ U24)'
```

```
** The final output of the model (drain current) is computed as the summation of
   contribution from each rule.
.param num='par(W1)*par(F1)+par(W2)*par(F2)+par(W3)*par(F3)+
   par(W4)*par(F4)+par(W5)*par(F5)+par(W6)*par(F6)+par(W7)*par(F7) +
   par(W8)*par(F8)+par(W9)*par(F9)+par(W10)*par(F10)+par(W11)*
   par(F11)+par(W12)*par(F12)+par(W13)*par(F13)+par(W14)*par(F14)+
   par(W15)*par(F15)+par(W16)*par(F16)+par(W17)*par(F17)+par(W18)*
   par(F18)+par(W19)*par(F19)+par(W20)*par(F20)+par(W21)*par(F21)+
   par(W22)*par(F22)+par(W23)*par(F23)+par(W24)*par(F24)'
.param den='par(W1)+par(W2)+par(W3)+par(W4)+par(W5)+par(W6)+
   par(W7)+ par(W8)+par(W9)+par(W10)+par(W11)+par(W12)+par(W13)+
   par(W14)+ par(W15)+par(W16)+par(W17)+par(W18)+par(W19)+
   par(W20)+ par(W21)+ par(W22)+par(W23)+par(W24)'
.param draincur='par(num)/par(den)'
** Define the subcircuit
.subckt dgmosfet n1 n2 n3
igate     n2     n3     0
gdrain n1     n3     cur='par(draincur)'
.ends
```

## References


[1] Y. Taur et al., "CMOS Scaling into the Nanometer Regime," *Proc. IEEE*, 1997, p. 486.

[2] L. Chang et al., "Extremely Scaled Silicon Nano-CMOS Devices," *Proc. IEEE*, 2003, p. 1860.

[3] T. Sekigawa and Y. Hayashi, "Calculated Threshold-Voltage Characteristics of an XMOS Transistor having an Additional Bottom Gate," *Solid State Electron.*, vol. 27, 1984, pp. 827-828.

[4] H.S. Philip et al., "Self-Aligned (Top and Bottom) Double-Gate MOSFET with a 25 nm Thick Silicon Channel," *Proc. IEEE*, 1997, p. 427.

[5] F. Djeffal et al., "An Approach Based on Neural Computation to Simulate the Nanoscale CMOS Circuits: Application to the Simulation of CMOS Inverter," *Solid-State Electron.*, vol. 51, 2007, pp. 48-56.

[6] F. Djeffal et al., "Design and Simulation of a Nanoelectronic DG MOSFET Current Source Using Artificial Neural Networks," *Materials Science and Engineering C*, vol. 27, 2007, pp. 1111-1116.

[7] S.C. Lo, Y. Li, and S.M. Yu, "Analytical Solution of Nonlinear Poisson Equation for Symmetric Double-Gate Metal-Oxide-Semiconductor Field Effect Transistors," *Mathematical and Computer Modelling*, vol. 46, 2007, pp. 180-188.

[8] J. He et al., "An Explicit Current-Voltage Model for Undoped Double-gate MOSFETs Based on Accurate Yet Analytic Approximation to the Carrier Concentration," *Solid-State Electron.*, vol. 51, 2007, pp. 179-185.

[9] H.C. Morris and A. Limon, "A Compact Model for the I-V Characteristics of an Undoped Double-Gate MOSFET," *Mathematics and Computers in Simulation*, vol. 79, no. 4, Dec. 2008, pp. 1116-1125.

[10] A. Cerdeira, B. Iniguez, and M. Estrada, "Compact Model for Short Channel Symmetric Doped Double-Gate MOSFETs," *Solid-State Electron.*, vol. 52, 2008, pp. 1064-1070.

[11] H. Lu, B. Yu, and Y. Taur, "A Unified Charge Model for Symmetric Double-Gate and Surrounding-Gate MOSFETs," *Solid-State Electron.*, vol. 52, 2008, pp. 67-72.

[12] F. Djeffal et al., "A Neural Approach to Study the Scaling Capability of the Undoped Double-Gate and Cylindrical Gate All Around MOSFETs," *Materials Science and Engineering B*, vol. 147, 2008, pp. 239-244.

[13] M. Hayati, M. Seifi, and A. Rezaei, "The Computational Intelligence in Simulation of DG MOSFET: Application to the Simulation of the Nanoscale COMS Circuit," *ICSE Proc., Johor Bahru*, Malaysia, 2008 , pp. 138-142.

[14] S. Datta, *Quantum Transport from Atom to Transistor*, 1st ed., Cambridge University Press, 2005, pp. 140-170.

[15] M. Lundstrom and J. Guo, *Nanoscale Transistors: Device Physics, Modeling and Simulation*, New York: Springer, 2006, pp. 55-120.

[16] Z. Ren and S. Goasguen, "NanoMOS 2.0 Source Code Download," Available: https://www.nanohub.org/resources/110/.

[17] R. Babuska, *Fuzzy Modeling for Control*, Norwell, MA: Kluwer Academic Publishers, 1998, pp. 90-150.

[18] J.S.R. Jang, C.T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, New Jersey: Prentice Hall, 1997, pp. 510-514.

[19] T. Takagi and M. Sugeno, "Fuzzy Identification of Systems and Its Applications to Modeling and Control, *IEEE Trans. Syst., Man, Cybern.*," vol. 15, no. 1, Feb. 1985, pp. 116-132.

[20] S.L. Chiu, "Fuzzy Model Identification Based on Cluster Estimation," *J. Intelligent and Fuzzy Systems*, vol. 2, 1994, pp. 267-278.

[21] J.S.R. Jang and C.T. Sun, "Neuro-Fuzzy Modeling and Control," *Proc. IEEE*, 1995, p. 378.

[22] H. Ben Hammouda et al., "Neural-Based Models of Semiconductor Devices for SPICE Simulator," *American J. of Applied Sciences*, vol. 5, 2008, pp. 385-391.

[23] Star-HSPICE Manual, Release 1998.4, Avant! Corporation, Fremont, CA, 1999.

**Mohsen Hayati** received the BE in electronics and communication engineering from Nagarjuna University, India, in 1985, and the ME and PhD in electronics engineering from Delhi University, Delhi, India, in 1987 and 1992, respectively. He joined the Electrical Engineering Department, Razi University, Kermanshah, Iran, as an assistant professor in 1993. At present, he is an associate professor with the Electrical Engineering Department, Razi University. He has published more than 60 papers in international and domestic journals and conferences. His current research interests include application of computational intelligence, artificial neural networks, fuzzy systems, neuro-fuzzy systems, and electronics circuit synthesis, modeling, and simulations.

**Majid Seifi** received the BS in electronics engineering from Sahand University of Techonology, Tabriz, Iran, in 2003, and the MS in electronics engineering from Razi University, Kermanshah, Iran, in 2009. He was with the Computational Intelligence Research Center in the Faculty of Engineering, Razi University during 2007 to 2009. He is currently working with the Iranian Offshore Oil Company. His current research interests include artificial intelligence, neural networks, fuzzy systems, neuro-fuzzy systems, and control systems.

**Abbas Rezaei** received the BS and MS in electronics engineering from Razi University, Kermanshah, Iran, in 2005 and 2009, respectively. He was with the Computational Intelligence Research Center, Faculty of Engineering, Razi University during 2007 to 2009. His current research interests include artificial intelligence, neural networks, fuzzy systems, and neuro-fuzzy systems.