Science Journal of Circuits, Systems and Signal Processing
2020; 9(1): 24-30
http://www.sciencepublishinggroup.com/j/cssp
doi: 10.11648/j.cssp.20200901.13
ISSN: 2326-9065 (Print); ISSN: 2326-9073 (Online)

Use of Virtual Forward Propagation Network Model to Translate Analog Components

Muhammad Sana Ullah[1], William Brickner[1], Emadelden Fouad[2]

[1]Department of Electrical and Computer Engineering, Florida Polytechnic University, Lakeland, USA
[2]Department of Natural Sciences, Florida Polytechnic University, Lakeland, USA

Email address:
mullah@floridapoly.edu (M. S. Ullah), williambrickner0186@floridapoly.edu (W. Brickner), efouad@floridapoly.edu (E. Fouad)

To cite this article:
Muhammad Sana Ullah, William Brickner, Emadelden Fouad. Use of Virtual Forward Propagation Network Model to Translate Analog Components. *Science Journal of Circuits, Systems and Signal Processing*. Vol. 9, No. 1, 2020, pp. 24-30. doi: 10.11648/j.cssp.20200901.13

Received: June 1, 2020; **Accepted**: June 17, 2020; **Published**: July 17, 2020

Abstract: Neural computing is an emerging research topic today due to its massive increase in demand and applications for machine learning. In this virtual simulation research work, using a free software, a program has been trained a neural network model and translate its functionality into the hardware. In the context of analog neural network, this research seeks to verify a shift sigmoid function that can approximate the transfer function of CMOS inverter. By showing this approximation accurately and reducing the number of components, it would help to implement the neural network based integrated chips. A conciliation is selected for the distance matric of the proposed function. This distance metric between the given CMOS transfer function and the shifted sigmoid function is minimized using the gradient descent. However, this approximate transfer function of CMOS inverter is chosen to verify in a three-layer perceptron networks. The network topology randomly generates weights to provide a diverse set of truth tables. We report two networks whose weights are chosen randomly using a back propagation algorithm due to volatile nature of the network topology and the activation function. The results of this research conclude that the transfer function of CMOS inverter is able to approximate the CMOS transfer function adequately for the purposes of these perceptron networks.

Keywords: Analog Components, Artificial Neural Network, Machine Learning, Universal Gates, Virtual Network

1. Introduction

Neural network is a data structure that used in machine learning, which are inspired by biology. Our ears lies the most power dense information-processing unit that known to science as the brain. If computers are able to do the processing either as efficiently or as quickly as the brain, then Von-Newman architectures would become outmoded [1, 2, 19]. The outside of neural networks of an analog forward propagation network is an example of a software algorithm. This algorithm is implemented in hardware and has an effect in trainable ASIC devices for applications such as autonomous vehicles, stock market predictions, and graphical rendering. The output of a neural network-training algorithm is a forward propagation network that is trained to approximate the complex function. If these networks are implemented in hardware, then this would be able to speed up the processing and even possibly introduce a new class of computer architecture [3, 4, 20].

Neurons in the brain are stimulated based on the activation of other neurons in the brain cell. When a brain cell receives enough stimulation from the adjacent brain cells, it will broadcast a signal to its neighboring neurons that will activate other neurons. This system of activating and associate neurons are the fundamental mechanism in what provides the functionality of the human brain [5, 6, 21, 22]. In software, this phenomenon is simulated as an artificial neural network (ANN). These cascade structure simulate the activation function. Besides, these can take on different functions without the need for changing the topology of the network and simulate using an ANN. In hardware, on the other hand, an artificial perceptron can recreate functionally by using operational amplifiers to add the incoming signals. After combining with a CMOS inverter, the perceptron would serve as an activation function and operational amplifier to buffer the output. Connections between neurons could weight by using resistors in conjunction with wires to span different values [5, 7, 21].

In this virtual simulation research work, we propose the idea that shows how a network of hardware components can recreate the distinct forward propagation networks with only changing the values of the resistances. To achieve this, firstly, a software simulation environment helps to abstract the hardware functionality in such a way that a network topology and weights could select using a standard backpropagation algorithm. Secondly, this trained network would put through a translation function that maps the weights to corresponding resistance values. After determining a feasible hardware configuration, thirdly, the abstraction can translate into a hardware tool to test the physics and accuracy of the proposed model.

The rest of the paper is organized as follows. Section 2 presents the proposed model and discusses the implementation technique systematically. Model accuracy and limitations are discussed in Section 3. Finally, Section 4 is followed by a conclusion.

2. Methodology and Implementation of the Proposed Model

The proposed model is a virtual representation of a mathematical model that is constructed to predict the observed values of an implementation of the corresponding analog forward propagation network. In this case, the function is chosen to reproduce an implementation of a universal NAND[1] logic gate and an arbitrary logic gate.

2.1. Activation Function Matching

Before creating a virtual network, an activation function has chosen which can sufficiently recreate the system function that occurs in its analog counterpart.

a) Choosing the Shape of Function: In the neural network, the sigmoid function is used to implement the differential properties of training network [6, 7]. The proposed activation function derives from the shape of a sigmoid function. However, it is shifted and stretched to resemble the transfer function of CMOS inverter by making the following transformations in (1).

$$A(x) = 1 - \frac{1}{1+e^{-\alpha(x-0.5)}} \quad (1)$$

To match the CMOS transfer function, the activation function is matched to the midpoint of the graph (shown in Figure 2) by subtracting 0.5 from the input x in the exponential. The most effective way to match the slope of the CMOS transfer function is to set the exponential function. By making the negative value, the inverting nature of a CMOS can be replicated by the sigmoid function. Specifically, the magnitude is determined by finding the geometric mean of minimized average of matching function and the minimized median of the activation function. This is accounting for the average of data but not accounting for the center of a dataset. Besides, it is rounded to the nearest integer. A sigmoid function is chosen over simulating the actual transfer function that is given by the Shockley diode equation. The Shockley diode equation is easier to recreate in the Wolfram Alpha API [9].

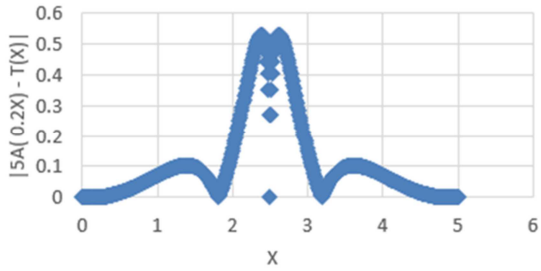


Figure 1. *The absolute value of the difference of the two function being matched over the spectrum of possible input values at α = 15.*

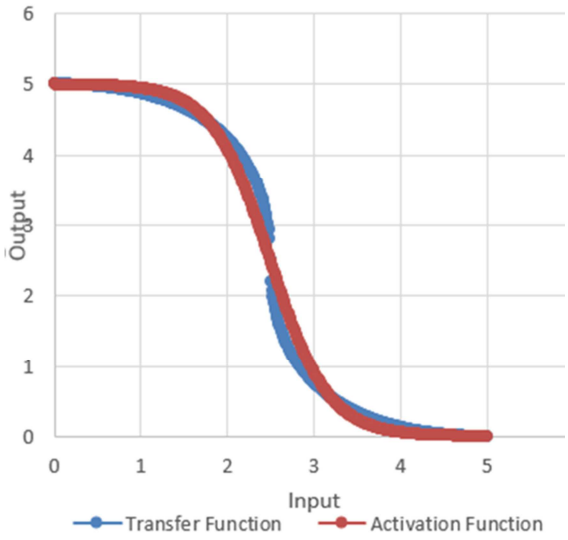


Figure 2. *A graph of the matched functions because of the gradient decent of the average difference for side-by-side comparison.*

Table 1. *Truth table Distributions for Candidate networks.*

Properties	Instances of truth table in the form F » f (0,0) f (0,1) f (1,0) f (1,1)			
Weights >.1 Figure 3	F » 0000 100000	F » 0001 0	F » 0101 0	F » 0100 0
	F » 0010 0	F » 0011 0	F » 0111 0	F » 0110 0
	F » 1010 0	F » 1011 0	F » 1111 0	F » 1110 0
	F » 1000 0	F » 1001 0	F » 1101 0	F » 1100 0
|Weights >.1| Figure 3	F » 0000 100000	F » 0001 0	F » 0101 0	F » 0100 0
	F » 0010 0	F » 0011 0	F » 0111 0	F » 0110 0
	F » 1010 0	F » 1011 0	F » 1111 0	F » 1110 0
	F » 1000 0	F » 1001 0	F » 1101 0	F » 1100 0
No Negative Weights	F » 0000 60206	F » 0001 1	F » 0101 4	F » 0100 32

[1] Even though a NAND gate can be implemented using a single perceptron, in order to test a prototype, there had to be at least 1 (one) hidden layer to observe the accuracy of forward propagation through hidden layers.

Properties	Instances of truth table in the form F » f (0,0) f (0,1) f (1,0) f (1,1)			
Figure 4	F » 0010	F » 0011	F » 0111	F » 0110
	31	3	13	15
	F » 1010	F » 1011	F » 1111	F » 1110
	62	5980	4623	1781
	F » 1000	F » 1001	F » 1101	F » 1100
	21125	1	57	6066
	F » 0000	F » 0001	F » 0101	F » 0100
	6156	651	1790	2623
Half Negative Weights Figure 4	F » 0010	F » 0011	F » 0111	F » 0110
	697	3257	4757	296
	F » 1010	F » 1011	F » 1111	F » 1110
	1752	5376	63248	1825
	F » 1000	F » 1001	F » 1101	F » 1100
	1790	224	2241	3317

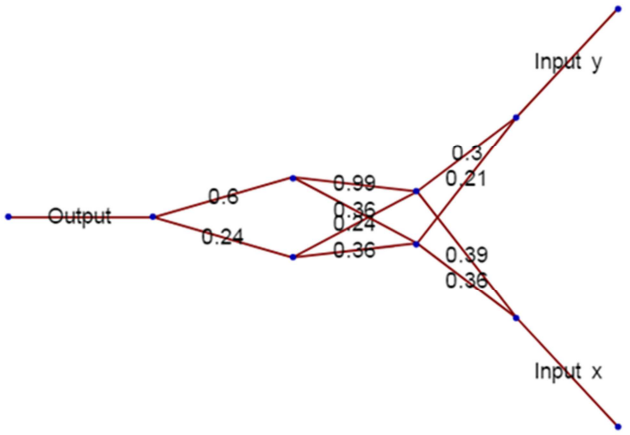


Figure 3. *The initial proposed network topology. Forward propagation N+1 layers with each layer having two nodes. After a sufficient amount for time the network failed to produce the desired output, after approximately 5 million times.*

$$M(x) = \left\{ f(x) \left| 5A\left(\frac{x}{5}\right) - A(x) \right| \right\} \tag{2}$$

In addition, the sigmoid function is easier to use in the back-propagation model due to their derivative output of a quadratic function.

b) Acquiring the CMOS Transfer Function: The method chosen for acquiring the transfer function is to export simulation data from the CMOS gate[2] into a spreadsheet. The advantage of the spreadsheet is that the large sets of data are easy to manipulate and help to find the center of a particular set of data. It is also a built-in feature of the spreadsheet platform.

c) Define Matching Function: In this case, we define the matching function by choosing an activation function of (1) which is the approximate transfer function, $A(x)$, over a given interval. Equation (2) can mathematically describe this matching function, $M(x)$, where $f(x)$ is the probability distribution of a choosing input. Matching function, $M(x)$ is optimized for minimizing the average that is shown in Figure 1. This provides data for the error distribution of all inputs. A side-by-side comparison of the matching function is displayed in Figure 2. This signifies that the activation function is a sufficient to approximate the transfer function.

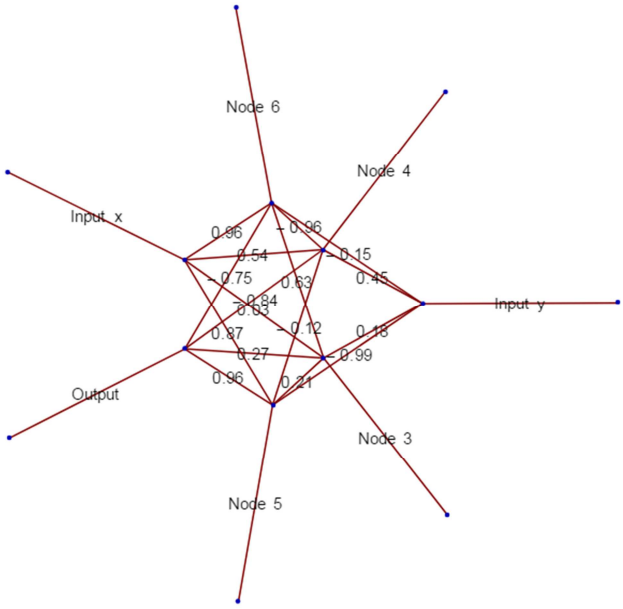


Figure 4. *The chosen network topology. The numbers on the connections represent the weights of the network being simulated. This network uses forward propagation to the N+1 layer, and the N+2 layer.*

d) Gradient Descent: To find the minimum iteration of matching function, $M(x)$ over the given interval, the gradient decent is chosen for its speed and low overhead programmatically. Gradient decent converges on a value by traversing an arbitrary function with a step size that is proportional to the slope of the function at a particular point. This chosen point converges a critical point when the step size of each iteration approaches zero[3]. This sequence converges due to the slope of the function being zero at local extrema (in this case the minimum). The sequence produced by this algorithm can be described by the following sequence[4].

$$M'(\mathrm{x}) = \frac{M(\mathrm{x}+h)-M(\mathrm{x})}{h} \tag{3}$$

To simplify $M'(x)$, the slope is approximated by using the formal definition of a derivative that shown in (3)[5].

2.2. Cloud Implementation of the Virtual Network

After determining an appropriate activation function, the proposed simulation is recreated the functionality of an analog network. To quickly iterate, the forward propagation of the analog network, a set of custom scripts are written in a free version of the Wolfram Cloud API [9]. The principle advantage of this custom script is to change the model so that

[2] The simulation recorded the output of a CMOS gate with inputs from 0 to 5 Volts, incrementing by 0.01 Volts.

[3] The interval is the operating range of 0-5 Volts.

[4] The distribution f(x) is assumed to be uniform.

[5] The parameter being optimized from Eq. 1 is α. α_0 is arbitrarily chosen to be 0.01 and X_o is arbitrarily chosen to be 13. ^{7}h is chosen to be the arbitrarily small value of 10E-13.

it could meet the specific growing needs of the project. The Wolfram Cloud API provides free native functionality that is useful for research such as advanced graphing and matrix multiplication.

2.3. Propagating Forward

The desired weights can be added to a matrix using a simple matrix multiplication to propagate the weights in the virtual network. These values could be propagated forward until an output is produced. The number of nodes depend on how these matrices multiply and what values multiply [8-10]. The connection of those nodes and strength of those connections are stored in matrices. The idea behind this project is that all of these values are determined when a network is trained for its purpose. A network can be treated like a black box and used for its intended purpose.

Table 2. *The Network Output.*

Logic Function	Input (V)		Virtual (V)	Analog (V)
Arbitrary	X: 0	Y: 0	4.9776	4.8862
	X: 0	Y: 5	0.0126	0.1948
	X: 5	Y: 0	4.1596	4.7837
	X: 5	Y: 5	0.0014	0.0961
NAND	X: 0	Y: 0	5.0000	5.000
	X: 0	Y: 5	4.9993	5.0000
	X: 5	Y: 0	5.0000	5.0000
	X: 5	Y: 5	0.0000	0.0000

2.4. Choosing a Propagating Network

After creating the scripts to preform matrix multiplication and the activation function, a network has selected so that it functions[6] are in a non-arbitrary manner.

a) Selecting Weights: In light of maintaining an active algorithm for determining a matrix, the weights on the network are randomly generated[7] by choosing a weight[8] using the distribution that shown in (4).

$$w = \frac{1}{x} \mid 1 < x < 10 \tag{4}$$

The truth table of the randomly generated network is verified against the desired output. If the generated network does not match the desired output, a new network generates until either a matching network is found, or the search lasted longer than 1 minute[9]. Since the network is small enough and the API ran fast enough, a brute force search is a viable option to test/verify ~10,000 networks per second.

b) Robustness: The first iteration of the neural network topology has one hidden layer, the second has two (see Figure 3[10]). For both iterations, a brute force search is run for a sufficiently long time that produce a set of weights with the desired truth table. To determine the ability of topology, a metric of a network topology is assigned the random weights to produce a desired truth table that has created. In light of this issue, the design of network has to change for a better solution. However, there is a need for a metric to evaluate an arbitrary topology before choosing an arbitrary network. The given name to this metric is robustness. Robustness is related to the ability of a particular network topology to produce a desired output by randomly assigning weights to the network and is measured as follows: a) select a network to test; b) generate an arbitrarily large number of truth tables; c) store the number of times each distinct truth table was observed; and d) examine the distribution, respectively.

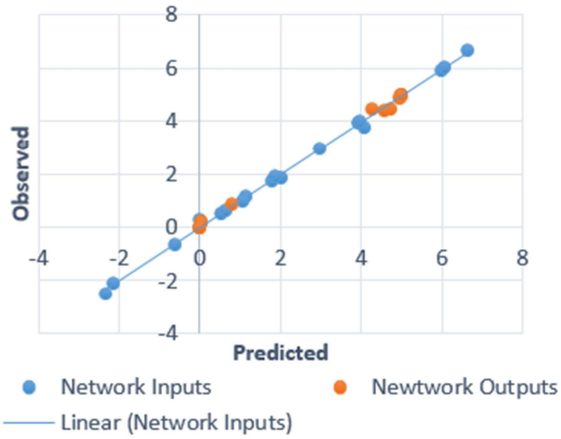


Figure 5. *Model Comparison with network inputs and outputs distribution.*

Some factors that drive into the selecting candidates for different topologies and weight assignments that are taking into account with the activation function has to be two stable states and map to each other. To prevent the layers from layers and always become alternating 1's and 0's, the topology should propagate each layer to the N+1 layer, and the N+2 layer (see Figure 4). The input propagates to the 1st and 2nd layer, and the 1st layer propagates to the second and third layer and so on. This would be estimated to continue as an activation function with M stable states where each layer propagates through the $N + 1$ layers. Typical truth table distribution of proposed network topology for different weights are shown in Table 1.

2.5. Virtual to Analog Translation

Multisim [11] is chosen to simulate the physics of the network that generates by the proposed model. The advantage of Multisim is that device physics can be accurately reproduced without the risk of breaking/damaging parts. In addition, the cost of making modifications to the circuit is negligible and the precision of components can be

[6] The desired functions are a NAND gate and an Arbitrary gate

[7] The distribution of X was even on the closed interval (1,10)

[8] This particular weight distribution was chosen because it reflects the inverse of the operation that the weight is being used for, multiplication

[9] One possible explanation for this is the range of values the weights that are allowed to take on but does not include zero (0).

[10] This number is chosen because the API kicks out the user if the run time of their script exceeds 1 minute and this is about how many times the check runs in 1 minute.

[11] It should be noted that LTspice [12] is a free circuit simulation software that can create networks in the same scope as was required for this project.

controlled.

a) Transfer Function: The model attempted to match an activation function is the transfer function of a CMOS inverter. In this case, we choose the CMOS inverter because it is power efficient. In addition, it is produced a symmetrical function. Therefore, the CMOS is matched for symmetry. The output of the CMOS has buffered so that the feedback loop of the OP-AMP does not feedback into the output of the CMOS.
b) Summation of Inputs Signals: To simulate the summation of weighted inputs, an OP-AMP summer is used in conjunction with differential OP- AMPs[12]. This allow to use the positive and negative weights for forward propagation of the perceptron. Table 2 shows the output of the proposed model for a virtual network verses the output of the analog network. Table 2 [13] shows the relationship of the values that is predicted by the proposed model and observed by the analog network simulation. Both Table 1 and Table 2 signifies that the virtual simulation is sufficient for estimating the output.

3. Accuracy and Limitations of the Proposed Model

The data that collect from the virtual network is used for a reproduction function. Data is collected from the analog simulation using voltage probes. A graph of predicted versus observed values (shown Figure 5) is created from the data that is collected. The correlation between the predicted inputs and observed inputs is 0.9990. The correlation between the predicted outputs and the observed outputs is 0.9992. Based on the data in Table 1, Table 2, and Figure 5, the difference between the proposed model and the observed network is almost negligible which validate the proposed model. Therefore, the proposed model is sufficient to accurately predict the function of the analog network in a small network using bounded values. The resources for this project are limited and cost of parts need to low as much as possible in all cases. If there are more resources available and any more work are to be done to extend the project, these would be following possible considerations.

3.1. Back Propagation

This project does not explore the prospect of back-propagation. This is due to the overhear associated with coding a back-propagation algorithm on an arbitrary network and is forgone in favor of a randomly generated network. Since a standard activation function is used to predict the transfer function of CMOS inverter, the virtual network could theoretically be trained using a standard back-propagation algorithm [10, 13, 21, 22]. This training would need to be tested for its rate of convergence, but other than there appears to be no reason why it would not be effective as the derivative which is still a linear combination of the output. If the virtual back propagation is not sufficient using the chosen virtual activation function, another possibility would be changed the slope of the analog transfer function by altering the attributes hardware components and then adjusting the virtual model accordingly. For analog back propagation development, it is possible that the derivative of the virtual activation function could be used to estimate the derivative of the transfer function using analog components. This could potentially be the first step in implementing a new analog back propagation algorithm.

3.2. Adjusting Resistances

In the scope of this project, we assumed the resistor with values of three significant figures existed and could reliably take on the values specified by the virtual to analog translation with high precision. This is obviously an issue when scaling due to the tolerance of resistors not being negligible. Resistors are also expensive space-wise in IC's. This would lead to a need to identify a component that would be able to adjust and store a resistance value such as a potentiometer or a memristor [14-15]. To be feasible as a programmable analog forward-propagation network, the logistics of storing and distributing resistance values coupled with the effects of noise on storing, setting, and distributing this component would have to be explored.

3.3. Scaling

The size of this network does not allow for the possibility of the deviation between the analog transfer function and the virtual activation function to be compounded. If the deviation is significant then the error could be reduced by virtually hashing. A linear approximation of the analog transfer function is implementing a back-propagation algorithm and selecting different weight ranges. In addition, scaling would create power consumption and speed that would need to be considered. Particularly OP-AMPs are not necessarily power efficient [13] and it might be possible to replace summers and difference amplifiers with CMOS [15]. It is important to change the way that could combine inputs of activation function which could be non-linear so that a different component could be used.

3.4. New Computer Architecture

If a programmable version of the forward propagation network could be implemented, then it is possible for some new computer architectures to be explored [3, 21, 22]. The idea here is that a chip could be included in the CPU which would be programmable. Network. Configurations for this analog network could be processed in software and then stored in a large repository that has to be accessed via the internet or to be stored locally. These functions not only determine theoretically but also locally. This programmable network would allow the designer an application specific

[12] This could theoretically be implemented with a single differential OP-AMP.
[13] The value (9.04, 6.04) was removed because it was an anomaly caused by the peak voltage of the OP-Amp only being 5V. As far as the CMOS is concerned, the difference between an input of 9V and 6V is negligible and raising the voltage too high can damage the CMOS.

integrated chip that could be included as a part of a software implementation [16, 18, 21].

4. Conclusion

This paper implemented a universal logic NAND gate and an arbitrary function using neural network. To match the sigmoid function with the transfer function, a few different distance metrics are chosen, and the distance is minimized using a gradient descent algorithm in a spreadsheet software. The spreadsheet software is used to simulate the values of a CMOS transfer function provided by hardware simulation software. Even though a few different metrics are pursued to minimize the distance, the changing of the chosen metric does not significantly affect the shape of the error function over the set of expected input values. Once a candidate sigmoid is selected, a set of networks are generated in software that is used the sigmoid as an activation function for the output. This perceptron network with sigmoid activation function is compared to a network that consist of op-amps for summation of the inputs, and CMOS for the transfer function of the network. This hardware is determined to be sufficiently close to the software for this particular size of perceptron network.

References

[1] T. Tuma, A. Pantazi, M. Gallo, A. Sebastian and E. Eleftheriou, "Stochastic phase-change neurons," *Nature Nanotechnology,* vol. 11, pp. 693-699, May 2016.

[2] C. D. Wright, "Phase-change devices: Crystal-clear neuronal computing," *Nature Nanotechnology*, vol. 11, pp. 655–656, May 2016.

[3] I. Aleksander, *Neural computing architectures: the design of brain-like machines*, London: North Oxford Academic, 1989.

[4] S. Furber and S. Temple, "Neural systems engineering," *Journal of The Royal Society Interface*, vol. 4, no. 13, pp. 193–206, 2006.

[5] H. P. Graf, L. D. Jackel and W. E. Hubbard, "VLSI Implementation of a neural network model," *Computer*, vol. 21, no. 3, pp. 41-49, March 1988.

[6] A. E. Pereda, "Electrical synapses and their functional interactions with chemical synapses," *Nature Reviews Neuroscience*, vol. 15, no. 4, pp. 250-263, April 2014.

[7] G. Gomes, T. Ludermir and L. Lima, "Comparison of new activation functions in neural network for forecasting financial time series," *Neural Computing and Applications*, vol. 20, no. 3, pp. 417-439, April 2011.

[8] B. M. Wilamowski, J. Binfet and M. O. Kaynak, "VLSI Implementation of Neural Networks," *International Journal of Neural Systems*, vol. 10, no. 3, pp. 191-197, June 2000.

[9] R. E. Maeder, *The Mathematica Programmer*, Academic Press, Inc., 1994.

[10] K. Hirasawa, M. Ohbayashi, M. Koga and M. Harada, "Forward propagation universal learning network," *IEEE International Conference on Neural Networks*, Washington, DC, USA, 3-6 June 1996.

[11] M. Jabri, S. Pickard, P. Leong, G. Rigby, J. Jiang, B. Flower and P. Henderson,"VLSI implementation of neural networks with application to signal processing," *IEEE International Symposium on Circuits and Systems*, pp. 1275-1278, 11-14 June 1991.

[12] X. Li, J. Qin, B. Huang, X. Zhang and J. B. Bernstein, "A new SPICE reliability simulation method for deep submicrometer CMOS VLSI circuits," *IEEE Transactions on Device and Materials Reliability*, Vol. 6, No. 2, pp. 247-257, June 2006.

[13] M. Valle, "Analog VLSI Implementation of Artificial Neural Networks with Supervised On-Chip Learning," *Analog Integrated Circuits and Signal Processing*, vol. 33, no. 3, pp. 263-287, December 2002.

[14] B. Vines and M. H. Rashid, "Memristors: The fourth fundamental circuit element," *IEEE International Conference on Electrical and Electronics Engineering*, Bursa, Turkey, 5-8 Nov. 2009.

[15] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu and W. J. Gross, "VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. PP, no. 99, pp. 1-12, 2017.

[16] V. Balan, "A low-voltage regulator circuit with self-bias to improve accuracy," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 2, pp. 365-368, Feb 2003.

[17] A. K. Shrinath, "Analog VLSI Implementation of Neural Network Architecture," *International Journal of Science and Research*, vol. 4, no. 2, pp. 653-656, February 2015.

[18] B. Razavi, *Design of Analog CMOS Integrated Circuits*, Second Edition, Mc Graw Hill Education, 2016.

[19] J. Cho, Y. Jung, S. Lee and Y. Jung, "VLSI Implementation of Restricted Coulomb Energy Neural Network with Improved Learning Scheme," *Electronics*, vol. 8, no. 563, pp. 1-13, May 2019.

[20] M. Yamaguchi, G. Iwamoto, H. Tamukoh and T. Morie, "An Energy-efficient Time-domain Analog VLSI Neural Network Processor based on a Pulse-width Modulation Approach," Computer Science, Emerging Technologies, Cornell University, pp. 1-13, February 2019.

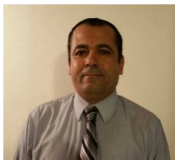
[21] Q. Wang, H. Tamukoh and T. Morie, "A Time-domain Analog Weighted – sum Calculation Model for Extremely Low Power VLSI Implementation of Multi-layer Neural Networks," Computer Science, Emerging Technologies, Cornell University, October 2018.

[22] S. Mada and S. Mandalika, "Analog Implementation of Artificial Neural Networks Using Forward Only Computation," *Asia Modelling Symposium (AMS)*, Kota Kinabalu, 4-6 December 2017, pp. 3-9.

Biography

Muhammad Sana Ullah (S'13-M'17) received the B. S. and M. S. degree in Electrical and Computer Engineering from Chittagong University of Engineering and Technology and Purdue University Northwest. Recently, he has finished his Ph.D. degree in Electrical and Computer Engineering from the University of Missouri-Kansas City and joined as an Assistant Professor of Electrical and Computer Engineering at Florida Polytechnic University. His specific research focuses are modeling of RLC interconnects and RF interconnect in high-density integrated circuits and investigation of a tunneling device based on graphene, carbon nano tube and other emerging 2D nanomaterials.

William Brickner (SM'17-) studied and recently graduate for the degree of Bachelor of Science in Computer Engineering at Florida Polytechnic University, Lakeland, FL. He is specialized on digital logic design. Now for his master's degree in electrical engineering, he attended the Colorado School of Mines where he is studying in the field of Electrical Engineering with Control Systems and Signal Processing. His research interest includes on virtual simulation network, power aware design, VLSI Design, analog neural network, digital systems design, control systems and signal processing. His dream is to make a fully functional analog neural network.



Emadelden Fouad received his B. Sc and M. Sc degree with Theoretical Physics from Cairo University, Egypt in 1996 and 2001 respectively. Emadelden also finished his PhD with theoretical Nano device from Cairo University, Egypt in 2005. Previously he worked as a teaching assistant and instructor with the department of Physics at Cairo University. Currently he is working as an assistant professor of physics with the department of natural science at Florida Polytechnic University. His research interest includes but not limited to quantum transport characteristics of energy efficient devices, Electromagnetic properties of type II superconductors and emerging nanomaterials.