Analog Integrated Circuits and Signal Processing (2021) 106:475–483
https://doi.org/10.1007/s10470-020-01713-x

Design and validation of an artificial neural network based on analog circuits

Fikret Başar Gencer[1] · Xhesila Xhafa[1] · Benan Beril İnam[1] · Mustafa Berke Yelten[1]

Received: 7 March 2020 / Revised: 7 March 2020 / Accepted: 5 September 2020 / Published online: 16 September 2020

Abstract
This paper focuses on the design and validation of an analog artificial neural network. Basic building blocks of the analog ANN have been constructed in UMC 90 nm device technology. Performance metrics of the building blocks have been demonstrated through circuit simulations. The weights of the ANN have been estimated through an automated back-propagation algorithm, which is running circuit simulations during weight optimization. Two case studies, the operation an XOR logic gate and a full adder circuit have been captured using the proposed analog ANN. Monte Carlo analysis of the XOR gate reveals that the analog ANN operates with an accuracy of 99.85%.

Keywords Artificial neural networks · Analog circuits · XOR gate · Full adder

1 Introduction

Artificial neural networks (ANNs) have originated by the effort to mimic the operation of a neuron. This has led to the generation of smart systems capable of learning from the available input–output samples of a system [3]. Through supervised learning in ANNs, multiple complex problems in various disciplines regarding optimization, classification, pattern recognition, and data fitting become easier to solve [11, 18, 23]. ANNs are mostly implemented in software based on the extensive capabilities of modern computing. Nevertheless, in many applications, problems regarding the computational complexity arise during implementation [8].

The structure of a typical ANN suggests that its operation can be greatly optimized when parallel processing can be adopted, which is also applicable when the ANN is implemented in hardware [4, 20, 22]. There have been analog, digital, and mixed-signal implementations of ANNs in the literature [15]. Among these approaches, analog implementations offer compliance with the continuous nature of ANNs, and they can be employed in sensor implementations, which enable analog signal processing without the need for analog-to-digital converters. The noise and the nonlinearity introduced by the analog circuits can be tolerated by the back-propagation (BP) that allows ANNs to operate correctly in the presence of computational errors [13]. Furthermore, analog circuits can be designed to be of low power and smaller area. This can potentially lead to energy-efficient implementations of analog neurons in comparison to digital neurons [16, 18, 25].

An analog ANN performs multiplication, subtraction, and summation along with the activation function evaluation. In our previous work, we developed the analog blocks of these basic operations and demonstrated the functionality of an analog ANN in UMC 90 nm CMOS technology [9]. In this paper, we extend the validation of our analog ANN on a full-adder. We also provide the process variations analysis of the analog ANN and show how its performance changes across the process corners of the device technology.

The organization of the paper can be provided as follows. Section 2 focuses on the building block design of the analog ANN. Then, in Sect. 3, the structure of the designed ANN is discussed. This is followed, in Sect. 4, by the introduction of the back-propagation algorithm and the training of the analog ANN through MATLAB-assisted circuit simulations. Case studies for validation will be

✉ Mustafa Berke Yelten
yeltenm@itu.edu.tr

[1] Electronics and Communications Engineering Department, Istanbul Technical University, Istanbul, Turkey

presented in Sect. 5, and subsequently, conclusions are drawn in Sect. 6.

2 Design of the circuit blocks

In order to understand the operation of the designed analog ANN, its building blocks have been described in this section. Circuit designs and corresponding results for the analog multipliers, the activation function circuitry and the current subtraction have been discussed and presented.

2.1 Analog multipliers

In the literature, different analog multiplier topologies have been introduced [1, 2, 5–7]. However, depending on the type of application, the specifications for each topology vary to some extent. In this paper, the classic Gilbert-cell multiplier has been used to ensure a four-quadrant multiplication for the first layer of the network [10]. Moreover, a variable gain amplifier (VGA) has been used as a second analog multiplier due to the connection requirements presented by other analog blocks in the network.

2.1.1 Gilbert cell-based multiplier

The first step to implement the analog ANN is to multiply the input signals with the corresponding weights. For this reason, a Gilbert cell-based multiplier operating with differential input signals (Fig. 1), has been used for the input layer of the ANN. The Gilbert multiplier is composed of three differential pairs whose working principles also define the DC behavior of the multiplier. In order to linearize the second-order current equation for the saturation mode of the CMOS transistors, the condition given in Eq. 1 should be satisfied,

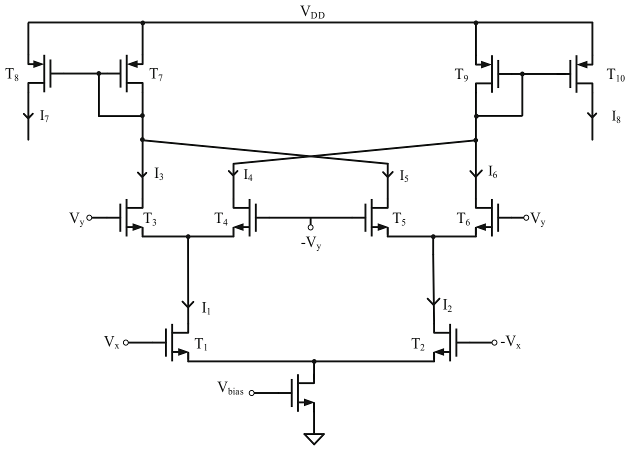


Fig. 1 The Gilbert cell-based analog multiplier

$$-\sqrt{\frac{I_{SS}}{K}} \leq V_{ID} \leq \sqrt{\frac{I_{SS}}{K}}, \tag{1}$$

where I_{ss} is the tail current and K is the linearity constant defined as $K = \mu_n C_{ox} \frac{W}{L}$ (A/V^2). Having satisfied this constraint, the resulting relationship between the output current (the difference of the differential output currents), as well as the input voltages V_x and V_y, is given by the following equation:

$$I_{out} = I_7 - I_8 = \sqrt{2} K V_x V_y \tag{2}$$

Analog signal multipliers are limited in their linear multiplication range due to the nonlinear nature of analog circuits. Hence, one of the most significant characteristics of an analog multiplier is the input range for linear multiplication. Figure 2 shows the input range, in which the Gilbert cell multiplier provides the most accurate behavior. For higher input values, the output current saturates. There have been several studies to increase the linear range of analog multipliers [17, 21]. Though a higher input range can be used for this particular application, it is mostly not necessary since the error in multiplication can be thought of as part of the nonlinear behavior of the ANN that will eventually be tolerated during the learning phase. By sweeping the values of V_x and V_y simultaneously, the linearity simulation results shown in Fig. 2 have been acquired.

2.1.2 Variable gain amplifier-based multiplier

Another way to realize an analog multiplier circuit is by using variable gain amplifiers. This topology is preferred as it is a two-quadrant multiplier. The output of the activation function is unipolar; hence, the topology shown in Fig. 3 is chosen for the hidden layers of the analog ANN.

The operation of a VGA leads to the output current expression in Eq. (3) that is similar to the expression of the Gilbert Cell-based multiplier.

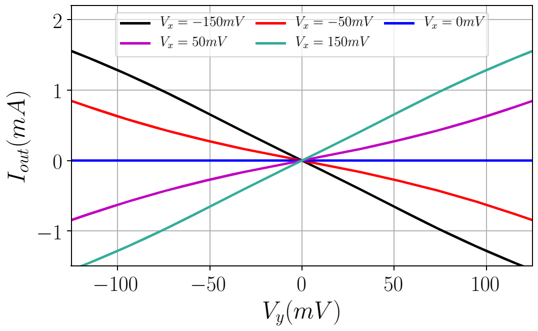


Fig. 2 The linear input range of the Gilbert cell-based multiplier

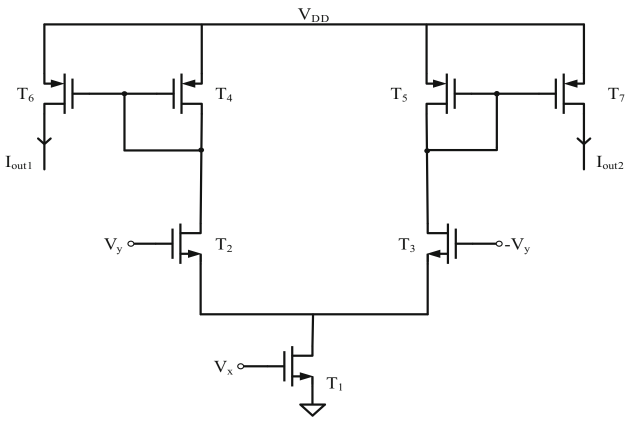


Fig. 3 The VGA-based analog multiplier

$$I_{out1} - I_{out2} = 2KV_xV_y \tag{3}$$

The designed VGA provides high linearity, as shown in Fig. 4. When the voltage V_y is changed, the current output changes proportionally to the change in voltage. To observe the transient output current results, the input voltage V_y is swept while the tail bias voltage input is kept constant. Thus, the output current is a scaled version of the multiplication of the two inputs.

2.2 Activation function circuitry

In ANNs, weighted sums are generated at the input of each neuron. If the corresponding neuron does not provide nonlinear mapping to the input, then the output expression of the ANN can be written as a linear function of the input. Although the linear input–output relationship could be adequate for simple problems such as common logic functions, most of the time, nonlinearity is required to implement ANNs for various other problems. Consequently, the need for an activation function arises. The choice of the activation function depends on how to evaluate the input signal of a neuron. The signal coming to the neuron input is passed through this weighted evaluation process to achieve the desired output. So, the shape of the activation function plays a role to decide whether the input signal of a neuron is worth to propagate to the next section or not. One of the most common activation functions in ANNs is the sigmoid function which resembles to a smooth step function.

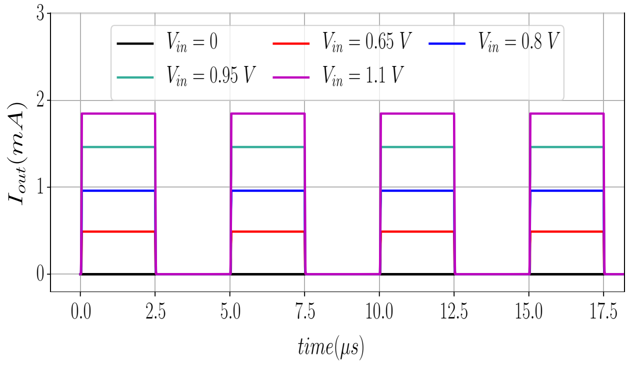


Fig. 4 The output current of the VGA

Another important characteristic of an activation function is that it should be differentiable within the specified input range, since the back-propagation algorithm requires the calculation of the gradient of the error with respect to the weights. Hence, the continuous and smooth input–output relationship of the sigmoid function enables a differentiable activation function, which is essential for back-propagation.

The circuit-based implementation of the sigmoid function is shown in Fig. 5 [14]. The input current of the circuit is mapped onto the voltage domain. The three characteristic regions of the sigmoid function have been emulated through the smooth current-voltage relationship of MOSFETs (Fig. 6). The input determines the operation region of T_1 and T_2. Negative input values yield small output voltages, and T_1 enters into the cut-off region. The same behavior also holds for T_2 around high input levels. The linear region of the curve is under the responsibility of T_3 and T_4. Finally, T_5 and T_6 are used to define the bias voltages of the previous four transistors.

2.3 Analog current subtraction circuit

As seen from the previous sections, analog multipliers have differential outputs, while the sigmoid function circuit requires a single current input. Hence, the need for current subtraction arises. The topology used to realize the latter is shown in Fig. 7. The working principle of this circuit is based simply on several current mirror structures. A major challenge in combining the analog circuit blocks in a neural network has been to ensure that the signal flows in the forward direction only. In this case, the output resistance at the subtraction point of the two differential

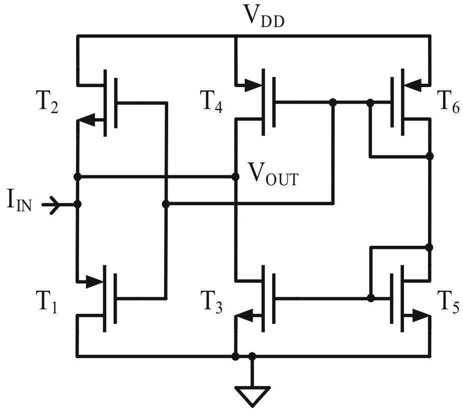


Fig. 5 The sigmoid generator circuit adapted from [14]

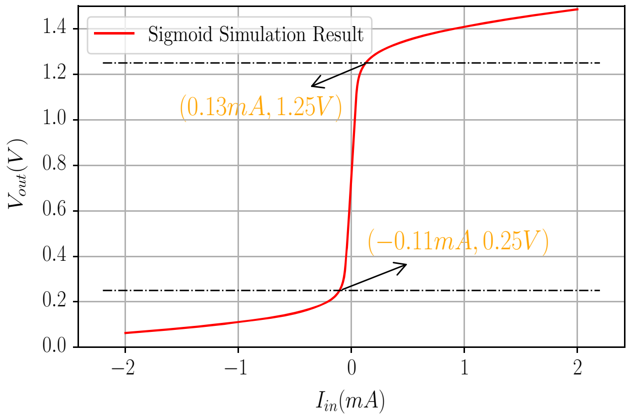


Fig. 6 The input–output relationship of the sigmoid generator circuit (dashed lines indicate the assumed logic levels)

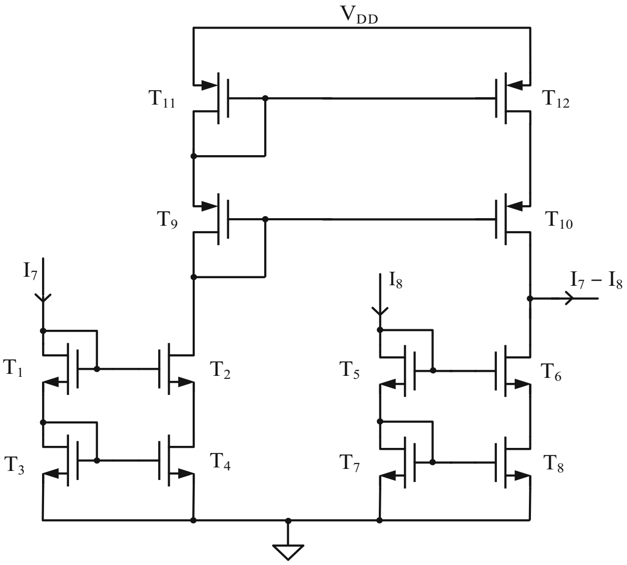


Fig. 7 The current subtraction circuit topology

currents has to be significantly high. Therefore, the current mirrors have been cascoded, such that the transconductance of the T_6 and T_{10} transistors will increase R_{out}, as described by Eq. (4).

$$R_{out} \cong g_{m_6} r_{o_6} r_{o_8} // g_{m_{10}} r_{o_{10}} r_{o_{12}}. \tag{4}$$

In order to correctly take into account the bias points of both analog multipliers, the same circuit topology has been used to design two current subtraction circuits, varying in terms of their aspect ratios.

3 Design of neural network layers

The simplest ANN unit perceptron has two inputs multiplied with the corresponding weights and its output is given to the input of the activation function. Basic logic gates such as AND, NAND, and NOR can be realized by using a single perceptron. Hence, the previously designed blocks are combined to come up with these logic gates. The Gilbert Cell-based multiplier takes two inputs and multiplies them with their weights. The activation function circuitry, which implements the sigmoid function decides if the output is whether 0 or 1. However, a simple perceptron may not solve more complex logic gates, such as the XOR gate. For this reason, a second layer, called the hidden layer, should be inserted between the input and output layers. To understand if the network requires hidden layers, the input–output relationship for the system of interest should be analyzed. If the system is linearly separable, the outputs can be then classified with a linear line. However, if the training data set is large, it may not be obvious whether the problem is linearly separable or not. As depicted in Fig. 8, the 2-2-1 network consists of two inputs, which are 0 or 1 for logic gates, one hidden layer consisting of two neurons, and finally a single output neuron.

4 The back-propagation algorithm

Based on its fundamental operations, i.e., multiplication, subtraction, addition, and the activation function, an ANN can be trained to learn from a given sample data set. For this purpose, weights should be introduced to the activation function, and will then be modified according to the convergence of the ANN outcomes to the actual sample outputs. Finding the correct weights is a challenge. Given that each neuron contributes to the overall accuracy of the results, a higher number of neurons would be beneficial; nevertheless, it also augments the complexity of finding the right values. The values that the weights can take vary in a significant range. The aim is to find the optimum combination of all weights for an ANN by incurring the least computational cost. One popular approach, called the back-propagation, aims to find the error between the ANN outcome and the actual sample output, which will be used toward adjusting the weights.

In the literature, several ANN learning techniques have been proposed to find an accurate weight set [24, 26]. In this work, the gradient descent back-propagation algorithm is selected to train the feed-forward network. Since the proposed ANN topology has no constraints on the learning performance, the chosen algorithm provides a simple and efficient implementation. The algorithm is developed in the MATLAB environment, and the input parameters are extracted from circuit simulations of the designed analog ANN. The system is fully automated, and all simulations can be performed in Cadence Analog Design Environment via Ocean script files, i.e., without involving the simulator

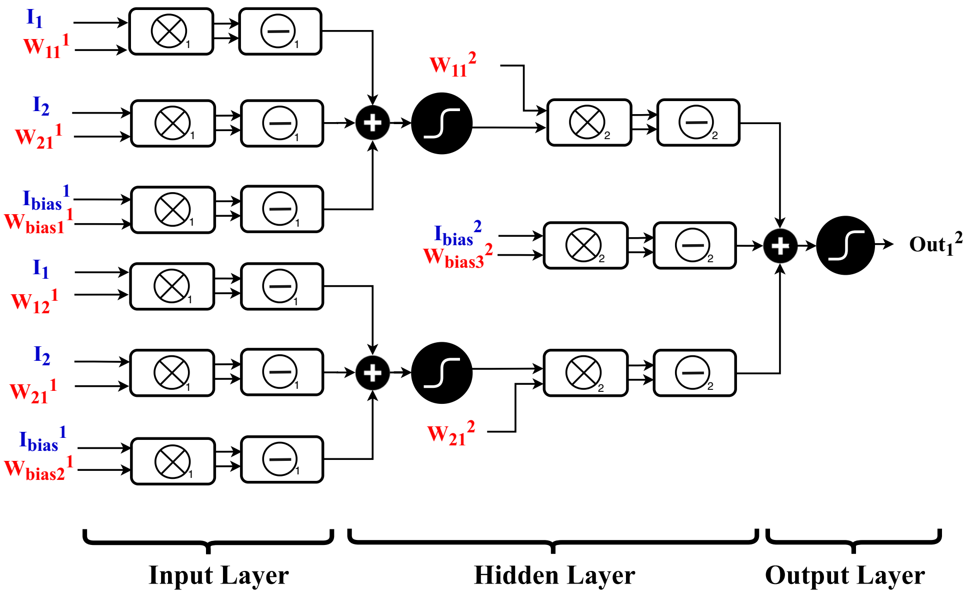
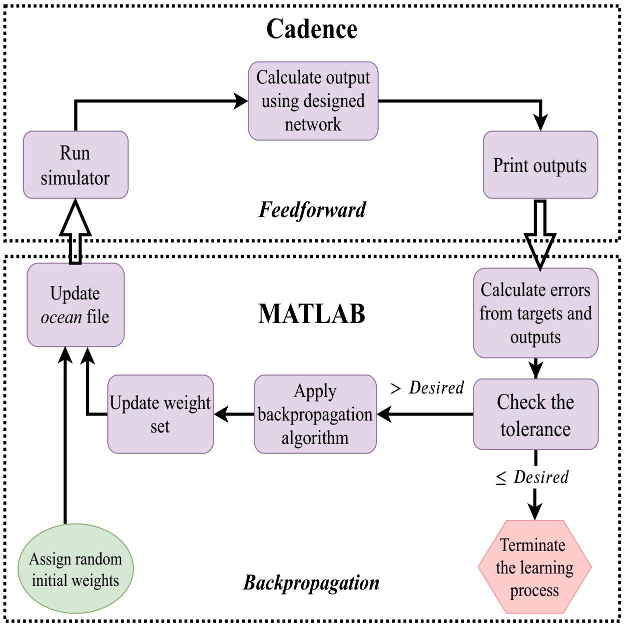


Fig. 8 The 2-2-1 network circuit block diagram

interface, which reduces the algorithm run-time. Figure 9 shows the implemented process steps.

The procedure starts by creating the Ocean files for the ANN system to operate correctly. Afterward, a random set of weights are sent to the circuit simulator in order to find the output values corresponding to that weight set. Simulation outcomes are then transmitted to the algorithm, which then calculates the error between the actual sample outputs and the ANN results.

The sum of squared errors (SSE) function is used as the cost function. If all outputs produced by the ANN for each sample satisfy the desired tolerance criteria, the program stops the learning process with the last weight set. If the error is larger than the tolerance, the back-propagation algorithm attempts to minimize SSE. In particular, the change in each weight is estimated by calculating the derivative of the cost function to the corresponding weight. Consequently, the derivative of the sigmoid plays a central role. The numerical derivative of the activation function from the output curve of the sigmoid circuit is calculated by the numerical finite difference method. After finding the change in each weight, the program updates the Ocean script. Subsequently, the circuit simulation is performed again with the newly calculated weights. As shown in Fig. 9, the flowchart loop runs until the tolerance criterion is reached, or the maximum iteration number, which has initially been set, is exceeded.



Cadence
Calculate output using designed network
Run simulator
Print outputs
Feedforward
MATLAB
Update ocean file
Calculate errors from targets and outputs
Update weight set
Apply backpropagation algorithm
> Desired
Check the tolerance
≤ Desired
Assign random initial weights
Backpropagation
Terminate the learning process

Fig. 9 The flowchart of the training and back-propagation algorithms

5 Case studies

The designed blocks have been utilized to initially demonstrate the operation of simple logic gates such as AND, OR, and NAND with a 2-1 network [9]. These gates are linearly separable and can be easily classified without the need for a hidden layer. Conversely, there are also logic gates that require at least one hidden layer, such as the XOR gate. The XOR gate has an input–output relationship that cannot be linearly classified. Therefore, to demonstrate the functionality of the implemented network, first, the functionality of an XOR gate has been demonstrated with the designed analog ANN. Subsequently, a circuit block with a topology of higher complexity, such as the full adder, has been implemented.

5.1 Case study I: The XOR gate

As the Boolean algebraic equation suggests in Eq. 5, the XOR gate can be composed of elementary logic gates such as the NAND, OR, and the AND gates. For the hidden layer of the network, the Gilbert Cell-based multiplier has been used to ensure a four-quadrant multiplication of the input voltages with the weights. In the output layer, the designed VGA realizes the multiplication of the single-ended output voltages of the sigmoid with the corresponding weights of this layer. Although the weights and the bias points found to mimic the NAND, OR, and AND gates are capable of solving the XOR problem, they have been recalculated using the back-propagation algorithm. The sigmoid-based activation circuit yields a maximum value close to 1.5 V and a minimum value tending near 0 V. In this work, an output level above 1.25 V indicates logic 1, while an output level below 0.25 V represents logic 0. The dashed lines in Fig. 6 illustrates this assumption. The signal provided as the input to the analog ANN is a continuous pulse signal with an amplitude of 70 mV for logic 1 and ground for logic 0. These voltage levels are sufficient to accomplish the output logic levels defined above. On the contrary, the output waveform is a pulse signal swinging between 750 mV and the logic value corresponding to the given inputs. The voltage level of 750 mV is set by the output of the sigmoid circuit with a zero input. Results produced by the analog ANN are shown in Fig. 10, which exactly correspond to the XOR truth table.

$$X \otimes Y = (X + Y)(\overline{XY}). \qquad (5)$$

5.2 Case study II: The full adder block

When the complexity of the system to be described by analog ANNs becomes larger, the nonlinear nature of analog circuits becomes a significant problem. In order to keep the accuracy of the analog ANN as high as possible, more neurons and additional hidden layers become necessary in the structure of the ANN [12]. To test the accuracy of the designed analog blocks and the analog ANN as a whole, a full adder circuit has been implemented. In Boolean algebra, a full adder can be built as a combination of simple logic gates, such as the AND, OR, and the NAND gate. Having described the XOR gate previously, the problem can be simplified, as shown in Fig. 11. In this topology, the analog ANN will have three inputs and two outputs with several hidden layers. Despite the increased complexity, the full adder truth table can be acquired by using the analog ANN where the previously found weights through back-propagation for the case study of XOR are leveraged. It can be seen in Fig. 12 that for all eight input combinations, the sum and the carry-out outputs of the analog ANN-based full adder are correctly determined, as in the XOR case study. The designed adder can also be modified to sum more than 1 bit, depending on the application.

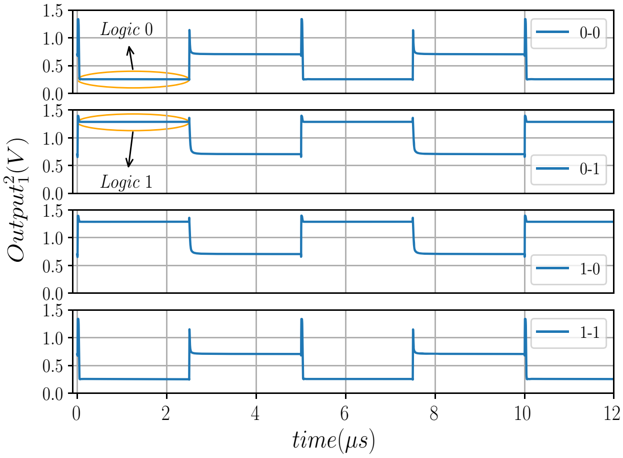


Fig. 10 The truth table representation of the XOR problem

5.3 Monte Carlo and process corner simulation results

Analog ANNs are subject to process variations that vary the values and the characteristics of the circuit components. Small deviations may result in acquiring erroneous outputs, thereby leading to a larger error [19]. To take into account the impact of these variations, the designed analog ANN is analyzed through Monte Carlo and process corner simulations. In the process corner analysis, in addition to the typical (tt) corner, four more corners have been simulated at the manufacturing conditions of fast NMOS—fast PMOS (ff), slow NMOS—slow PMOS (ss), slow NMOS—fast PMOS (snfp), and fast NMOS—slow PMOS (fnsp). The histograms of logic 1 (right) and logic 0 (left) values depicted in Fig. 13 have been achieved by the Monte Carlo simulation of 1000 samples each. According to these histograms, it can be seen that logic 0 can significantly vary, whereas logic 1 is much more constrained in its distribution. If any value below 1.25 V can be assumed to be logic

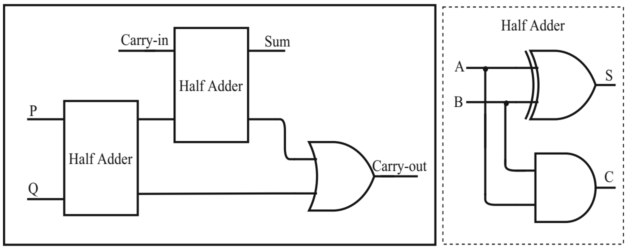


Fig. 11 The full adder topology

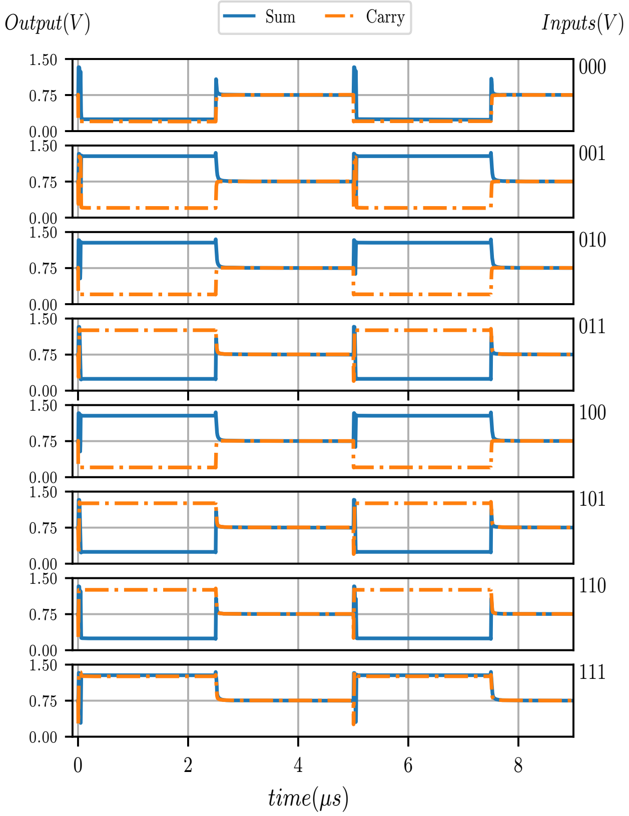


Fig. 12 The truth table representation of the full adder

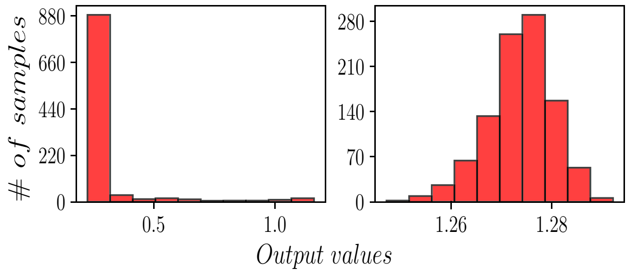


Fig. 13 Monte Carlo simulation results

0, and above it logic 1, then the performance accuracy of the XOR gate based on the analog ANN can be found to be 99.85% (i.e. 3 incorrect logic evaluation of the XOR gate out of 2000 samples).

The process corner analysis shown in Table 1 verifies that the output logic values are similar to the results obtained by using the typical (tt) process corner of transistors. Additionally, it can be seen that the logic 1 limit decreases, and the logic 0 limit increases for the ff process corner. These small changes can be properly taken into account by re-training the weights through back-propagation. Consequently, a new weight set that better matches the observed results of the analog ANN to the design specifications can be found. So, it can be concluded that analog ANNs can tolerate the manufacturing inaccuracies and swiftly adapt to new conditions.

Table 1 Process corner analysis results

Case	tt	ss	snfp	fnsp	ff
Logic 1	1.28	1.341	1.266	1.297	1.214
Logic 0	248.2	250.6	261.8	244.8	272.8

6 Conclusion

In this paper, an analog ANN, which can solve linearly separable and non-linearly separable problems, has been designed. The operation of the analog ANN has been described along with its circuit blocks in detail. The designed analog ANN requires a set of weight values, which need to be optimized based on the samples provided for training. To determine the optimum weight values, supervised learning, along with back-propagation, has been employed in an automated environment involving an algorithm and a circuit simulator. The number of inputs and hidden layers of the analog ANN can be increased depending on the complexity of the problem. In order to test the accuracy of the analog ANN, an XOR gate and a full adder circuit have been successfully implemented, where the XOR gate is also applied to Monte Carlo and corner analysis. It has been verified that the analog ANNs are relatively robust to manufacturing variations. Considering the small size of analog ANNs, medical or internet of things applications can be some of the fields, in which analog ANNs will be able to contribute to the on-chip learning process.

References

1. Akshatha, B. C., & Akshintala, V. K. (2009). Low voltage, low power, high linearity, high speed CMOS voltage mode analog multiplier. In *2009 2nd international conference on emerging trends in engineering technology* (pp. 149–154). https://doi.org/10.1109/ICETET.2009.141.
2. Aksin, D. Y., Basyurt, P. B., & Uyanik, H. U. (2009). Single-ended input four-quadrant multiplier for analog neural networks. In *2009 European conference on circuit theory and design* (pp. 307–310). https://doi.org/10.1109/ECCTD.2009.5274983.
3. Bai, Y., Fan, D., & Lin, M. (2018). Stochastic-based synapse and soft-limiting neuron with spintronic devices for low power and robust artificial neural networks. *IEEE Transactions on Multi-Scale Computing Systems*, *4*(3), 463–476. https://doi.org/10.1109/TMSCS.2017.2787109.
4. Bayraktaroglu, I., Ogrenci, A. S., Dundar, G., Balkir, S., & Alpaydin, E. (1997). An analog neural network synthesis system.

In *Proceedings of international conference on neural networks* (pp. 910–915).

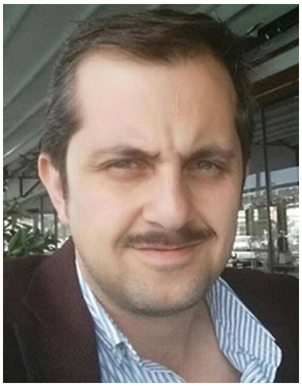
5. Chen, Z., Zheng, Y., Choong, F. C., & Je, M. (2012). A low-power variable-gain amplifier with improved linearity: Analysis and design. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *59*(10), 2176–2185. https://doi.org/10.1109/TCSI.2012.2185331.
6. Chen, C., & Li, Z. (2006). A low-power cmos analog multiplier. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *53*(2), 100–104. https://doi.org/10.1109/TCSII.2005.857089.
7. Colli, G., & Montecchi, F. (1996). Low voltage low power CMOS four-quadrant analog multiplier for neural network applications. In *1996 IEEE international symposium on circuits and systems circuits and systems connecting the world. ISCAS 96* (Vol. 1, pp. 496–499). https://doi.org/10.1109/ISCAS.1996.539993.
8. Du, Y., Du, L., Gu, X., Du, J., Wang, X. S., Hu, B., et al. (2018). An analog neural network computing engine using CMOS-compatible charge-trap-transistor (CTT). *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. https://doi.org/10.1109/TCAD.2018.2859237.
9. Gencer, F. B., Xhafa, X., İnam, B. B., & Berke Yelten, M. (2019). Design of ananalog circuit-based artificial neural network. In *2019 11th international conference on electrical and electronics engineering (ELECO)* (pp. 379–383). https://doi.org/10.23919/ELECO47770.2019.8990559.
10. Gilbert, B. (1968). A precise four-quadrant multiplier with subnanosecond response. *IEEE Journal of Solid-State Circuits*, *3*(4), 365–373.
11. Giordano, M., Cristiano, G., Ishibashi, K., Ambrogio, S., Tsai, H., Burr, G. W., et al. (2019). Analog-to-digital conversion with reconfigurable function mapping for neural networks activation function acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, *9*(2), 367–376. https://doi.org/10.1109/JETCAS.2019.2911537.
12. Haykin, S. O. (2009). *Neural networks and learning machines* (3rd ed.). New York: International Edition.
13. Jain, A. K., Jianchang, M., & Mohiuddin, K. M. (1996). Artificial neural networks: a tutorial. *Computer*, *29*(3), 31–44.
14. Khodabandehloo, G., Mirhassani, M., & Ahmadi, M. (2012). Analog implementation of a novel resistive-type sigmoidal neuron. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *20*(4), 750–754.
15. Merkel, C., Kudithipudi, D., & Sereni, N. (2013). Periodic activation functions in memristor-based analog neural networks. In *The 2013 international joint conference on neural networks (IJCNN)* (pp. 1–7). https://doi.org/10.1109/IJCNN.2013.6706772.
16. Moon, S., Shin, K., & Jeon, D. (2019). Enhancing reliability of analog neural network processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *27*(6), 1455–1459.
17. Nandini, A., Madhavan, S., & Sharma, C. (2012). Design and implementation of analog multiplier with improved linearity. *International Journal of VLSI Design and Communication Systems*, *3*(5), 93.
18. Ngwar, M., & Wight, J. (2015). A fully integrated analog neuron for dynamic multi-layer perceptron networks. In *2015 international joint conference on neural networks (IJCNN)* (pp. 1–8). https://doi.org/10.1109/IJCNN.2015.7280448.
19. Orgenci, A. S., Dundar, G., & Balkur, S. (2001). Fault-tolerant training of neural networks in the presence of MOS transistor mismatches. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *48*(3), 272–281. https://doi.org/10.1109/82.924069.
20. Satyanarayana, S., Tsividis, Y. P., & Graf, H. P. (1992). A reconfigurable VLSI neural network. *IEEE Journal of Solid-State Circuits*, *27*(1), 67–81. https://doi.org/10.1109/4.109558.
21. Soo, D. C., & Meyer, R. G. (1982). A four-quadrant NMOS analog multiplier. *IEEE Journal of Solid-State Circuits*, *17*(6), 1174–1178.
22. Vittoz, E. A. (1990). Analog VLSI implementation of neural networks. In *IEEE international symposium on circuits and systems* (Vol. 4, pp. 2524–2527). https://doi.org/10.1109/ISCAS.1990.112524.
23. Yelten, M. B., Franzon, P. D., & Steer, M. B. (2012). Comparison of modeling techniques in circuit variability analysis. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, *25*(3), 288–302. https://doi.org/10.1002/jnm.836.
24. You, Z., & Lu, C. (2018). A heuristic fault diagnosis approach for electro-hydraulic control system based on hybrid particle swarm optimization and Levenberg–Marquardt algorithm. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-018-0962-5.
25. Zamanlooy, B., & Mirhassani, M. (2017). An analog CVNS-based sigmoid neuron for precise neurochips. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *25*(3), 894–906.
26. Zhou, P., & Austin, J. (1998). Learning criteria for training neural network classifiers. *Neural Computing and Applications*, *7*(4), 334–342.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Fikret Başar Gencer received his B.Sc. degree in electrical engineering from Istanbul Technical University (İTÜ) Electronics and Communications Engineering Department, Turkey, in 2019, and his B.Sc. degree in physics from İTÜ Physics Engineering Department in 2020. He is currently working toward his M.Sc. degree in Electronics Engineering at İTÜ Electronics and Communications Engineering Department. His research interests concentrate on machine learning, analog circuit design, and semiconductor device modeling.

Xhesila Xhafa received her B.Sc. degree in electrical engineering from Istanbul Technical University (İTÜ) Electronics and Communications Engineering Department, Turkey, in 2019. She is currently working toward her M.Sc. degree in Electronics Engineering at at İTÜ Electronics and Communications Engineering Department, where she also works as a researcher in the İTÜ VLSI laboratory. Her research topics include analog and RF circuit design, analog neural networks and surrogate modeling.

Benan Beril İnam received her B.Sc. degree in in electrical engineering from Istanbul Technical University (İTÜ) Electronics and Communications Engineering Department, Turkey, in 2019. She is currently working toward her M.Sc. degree in Electronics Engineering at at İTÜ Electronics and Communications Engineering Department and working as a researcher in the İTÜ VLSI Laboratory. Her research interests concentrate on analog/RF integrated circuit design.



Mustafa Berke Yelten (Senior Member, IEEE) received the B.Sc. degree (High Hons.) in electrical engineering from Bogaziçi University, İstanbul, in 2006, and the M.Sc. and Ph.D. degrees in electrical engineering from North Carolina State University, Raleigh, in 2008 and 2011, respectively. He was a Quality-Reliability Research Engineer with Intel Corporation, Hillsboro, Oregon, from 2011 to 2015. In 2015, he joined the Electronics and Communications Engineering Department, Istanbul Technical University, where he serves as an Associate Professor since 2020. His research interests include the design, optimization, and modeling of nanoscale transistors and the design of analog/RF integrated circuits.