Analog Performance Locking through Neural Network-Based Biasing

Georgios Volanis, Yichuan Lu, Sai Govinda Rao Nimmalapudi,
Angelos Antonopoulos, Andrew Marshall and Yiorgos Makris
Department of Electrical and Computer Engineering, The University of Texas at Dallas

***Abstract*—We introduce a method for protecting analog Integrated Circuits (ICs) against unauthorized use by obfuscating their operating point using an analog neural network. With the model of the trained analog neural network acting as a lock and its inputs as the key, only the correct key combination will unlock the analog IC, by providing it with the required bias conditions to operate within its specification limits. By defining the key combinations in the continuous analog space and by using floating gate transistors to realize the neural network, the proposed method defends itself against efforts to guess the correct key through model approximation attacks. Moreover, by inhibiting retraining of the analog neural network, the proposed solution enables customization of the lock and key combination to each IC. The proposed solution has been implemented in silicon through a proof-of-concept experimental setup comprising a Low-Noise Amplifier (LNA) and a programmable analog neural network. Experimental results demonstrate the effectiveness of the method in preventing unauthorized use of an analog IC.**

I. INTRODUCTION

The increasing use of integrated circuits (ICs) in various industrial sectors, including telecommunications, automotive and military systems, is accompanied by a rising level of concern regarding IC reliability and security. Prevention of unauthorized use and recycling of such ICs constitute two major challenges in the field of hardware security. Indeed, many mission-critical ICs include sensitive information which should not be disclosed to unauthorized parties. Moreover, IC recycling costs semiconductor industry billions of dollars annually in lost revenue. As a result, several methods have been proposed for locking ICs against unauthorized usage.

The majority of such locking techniques published so far focus on the protection of digital ICs. In most of these works [1], [2], [3], [4] a combinational logic locking technique is applied by leveraging the large number of transistors to hide functionality. Digital designs are obfuscated by adding extra inputs and logic gates, which prevent an IC from implementing the correct Boolean function unless a correct key is provided at these inputs. Locking of analog ICs, on the other hand, is a significantly different and more challenging task. This is mainly due to limited number of topologies for implementing typical analog blocks, as well as the inherently small number of transistors and the consideration of multiple parameters during the design process. For this reason, hiding IC functionality is not a plausible path for locking analog ICs. However, locking analog IC performance is a potential solution to this problem. Specifically, the idea behind this approach is to require a secret key so that an analog block operates within its specifications.

Two recently proposed methods [5], [6] follow the aforementioned analog performance locking paradigm. In both of these solutions, the idea of combinational locking of biasing currents in analog circuits is explored. In [5], the bias is obfuscated by using multiple branches of biasing transistors. The number of on-transistors is controlled by a digital key and only the correct key can bias the circuit to the desired operating point. The authors of [6], combine the idea of hardware metering and bias locking to create a unique key for each chip. While these method offer the first solutions for analog design obfuscation, they are susceptible to the lock removal attack. Furthermore, in the solution of [5], the relation between the key and the IC performance is simple and can be learned by model approximation attacks. In [6], however, this limitation is ameliorated through the use of satisfiability modulo theories for designing a configurable current mirror. Finally, in both these works, the key is in the digital domain whereas we are interested in exploiting the continuous analog space for defining keys.

In this work, we propose a method for protecting analog ICs from unauthorized use, by locking their performance through an analog neural network. This is achieved by eliminating direct access to the biasing inputs of the analog IC and by relying, instead, on the analog neural network for biasing the IC to its operating point. Essentially, the trained neural network acts as a lock that provides the desired bias voltages to the IC if and only if the analog correct key is applied at its inputs. In other words, only the input of the neural network that corresponds to the correct key will result in the operation of the IC within its specification limits. Programmability of the analog neural network through the use of analog floating gate transistors (FGTs), which serve as permanent storage for the synapse weights, along with the continuous input domain, which allows for a very large number of analog key options, make our method secure against brute force and model approximation attacks for guessing the key. Additionally, they enable individualization of the key per chip, which further strengthens the proposed method.

The proposed method is demonstrated in silicon using an LNA chip [7] and a programmable analog neural network prototyping chip [8]. Experimental results corroborate robustness of the proposed analog performance locking method against both brute-force and model approximation attacks.

The remainder of the paper is structured as follows. In Section II, we discuss the threat model considered by our locking method. In Section III, we discuss the role of biasing in analog ICs and we introduce our neural network-based biasing method for analog performance locking along with a metric for evaluating its effectiveness. Our experimental platform and our obtained results are presented in Section IV and Section V, respectively, and conclusions are drawn in Section VI.

II. THREAT MODEL

In the threat model considered in this work, the adversary is an unauthorized user who has gained illegitimate access to an

978-1-7281-1170-4/19/$31.00 ©2019 IEEE

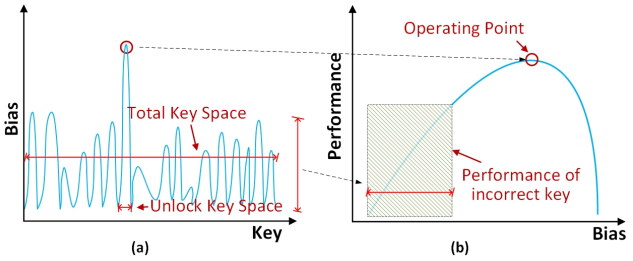

Fig. 1. (a) Proposed Function for Obfuscation (b) Performance VS. Bias

analog IC. For example, this could be a previously used and recycled IC, or an IC possessing sensitive capabilities, such as in military applications, which has ended up in the wrong hands. The objective of the proposed method is to prevent the unauthorized user from operating the illegitimately accessed IC within its specifications, by requiring an analog key for unlocking its performances. The adversary is aware of the fact that a locking mechanism exists and is allowed to experiment non-invasively with key values. The adversary is also allowed to legitimately acquire another copy of this IC and its key.

The proposed method is not intended towards protecting the intellectual property (IP) of a design from unauthorized overproduction, cloning or lock removal by an untrusted foundry which has access to the layout of the design.

III. Performance Locking in Analog ICs

In this section, we first discuss the importance of biasing for proper operation of an analog IC and we explain why these biases are insufficient to serve as keys. We, then, introduce the proposed analog performance locking method along with a metric for assessing its effectiveness.

A. Analog Performance Locking Through Biases

In an analog circuit, biasing is the action of providing carefully selected voltages or currents, which are called biases, at various nodes and terminals of the circuit. During the design process, bias values are determined along with other parameters such as transistor dimensions, to produce the desired performance. As shown in Figure 1(b), provision of the chosen biasing values brings the circuit to its operating point, wherein it exhibits its optimal performance. A small change in the biases will drive the circuit out of its operating point, thereby causing performance degradation or even specification failure. In other words, the provided biases are a critical part of the design since they can effectively vary the performance of the circuit [7].

A straightforward idea for locking the performance of an analog IC would be to treat the biases as the key, since only the correct bias values would result in acceptable performance. Such an approach, however, is relatively easy to defeat. The reason for this is that the relation between biases and performance are fairly easy to learn through model approximation algorithms, such as Powell′s method [9] and Gradient Minimization [10]. Indeed, as we will show experimentally in Section V.A, such methods require a very small number

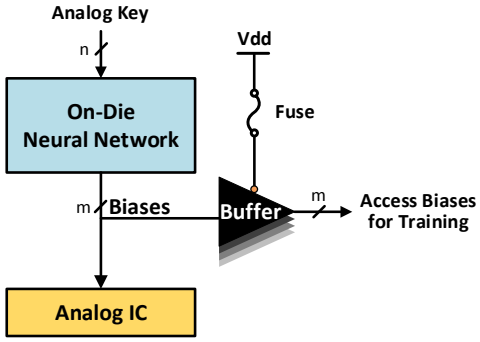

Fig. 2. General Architecture of the Proposed Locking Method

of samples to deduce the optimal biases for a circuit. Furthermore, an experienced designer can estimate the proper biases and assist model approximation attacks by initializing the sampling process closer to the operating point. Therefore, using biases as keys is a weak performance locking strategy.

B. Proposed Analog Performance Locking

Idea & Architecture: In this work, we propose a method for locking the performances of an analog IC by preventing direct application of biases. Rather, in order to provide the biases required for specification-compliant IC operation, we rely on an on-die analog neural network, as shown in the architecture depicted in Figure 2. The analog neural network is trained to implement a function[1] similar to the one shown in Figure 1(a). In essence, the trained analog neural network serves as a lock, while its analog input serves as the key. A correct pair of lock and key results in generation of the biases needed for bringing the IC to its operating point. Any other combination, however, should result in biases for which the performance of the IC does not meet its specifications. Moreover, incorrect combinations should minimize leakage of any information that could be leveraged by model approximation algorithms to discover the operating point of the analog IC.

Lock Implementation Protocol: The proposed method seeks to protect against unauthorized use of a circuit by anyone other than the legitimate end-user. To this end, before distributing an IC to its end-user, our method programs the analog neural network to accept a specific analog key. The model learned by the neural network is permanently stored on-chip through the use of analog FGTs which act as permanent synapse weight storage. The analog key is, then, provided to the end-user along with the IC. The end-user must apply this key to the IC in order to provide the correct biases and unlock the IC performance. Given the continuous nature of the analog input space and the fact that the neural network model is programmed after chip fabrication, each IC can be programmed to accept its own unique key.

Key Application & Storage: Several options exist for applying and storing the analog key to unlock an IC. The

[1]Neural networks are powerful machine learning entities which, based on the universal approximation theorem, can approximate any continuous function.

simplest approach is to directly provide these analog values through external pins. Another approach, similar to locking methods for digital ICs, is to use static random-access memory (SRAM) cells alongside a digital to analog converter (DAC) and sample-&-hold circuits with a refresh mechanism to complete the scheme. Alternatively, the analog key can be directly programmed as charge in analog floating gate transistors at power-up and removed at power-down. While analog FGTs are more challenging to program, this solution keeps the proposed method entirely in the analog domain, and eliminates the need for SRAM, DAC and sample-&-hold circuits.

Attack Resistance: The ability of the proposed method to withstand attacks is based on four facts. First, the size of the analog input space and the time-consuming process of applying and evaluating candidate analog keys prohibits brute force attacks. Second, due to the high entropy of the function learned by the neural network (see Figure 1(a)), when incorrect analog keys are applied the resulting analog IC performance does not provide guidance towards the correct key. Third, as mentioned earlier, each IC has its own key, hence key sharing is not a viable attack. Lastly, as shown in Figure 2, once the neural network is programmed, observability of its output (i.e., bias voltages) which is needed for the purpose of training, is eliminated by blowing a fuse.

C. Attack Difficulty Metric

In order to quantify the difficulty encountered by an attacker while attempting to deduce the analog key, we need to define an appropriate metric. Using Figure 1(a) as a reference, let N_s be the magnitude of the space wherein the analog key draws values from. Also, let N_p be the number of options in N_s which result in biases that either bring the circuit to its operating point or provide significant information (e.g., performances close to the specifications) which can assist model approximation attacks. Then, the probability of deducing the analog key can be quantified as:

$$metric = N_p/N_s \tag{1}$$

IV. EXPERIMENTAL PLATFORM

To demonstrate the proposed analog performance locking method, we emulated its implementation using an experimental platform which consists of two custom-designed and fabricated ICs, namely an LNA chip with three bias voltages and an analog neural network prototyping chip. Details of these two ICs are provided below.

A. LNA with Bias Voltages

The LNA, which is the circuit that we want to protect against unauthorized use, was designed and fabricated in GlobalFoundries' 130nm RF CMOS process. Its schematic and specifications are shown in Figure 3. It is a cascode LNA with inductive source degeneration that consists of only three transistors. M1 is used as a common source transistor, M2 is a common gate transistor whereas M3 is used for bias purposes. Transistors M1, M2 are biased by using voltages Bias1 and Bias2 respectively while Bias3 is the supply voltage of our LNA. These three bias voltages must be set to exact values in order for the LNA to operate within its specification limits [11]. In the proposed analog performance locking method, instead of being supplied externally, these three bias voltages must be generated by an analog neural network if and only if the correct analog values (i.e., key) are provided at its inputs.

B. Analog Neural Network Platform

The analog neural network, which is used to lock the performances of the LNA, is implemented by using our custom-designed neural network experimentation platform. The block diagram of the chip, which was fabricated in TSMC's 350nm process [8], is shown in Figure 3. The core is a reconfigurable 30x20 array of synapses (S) and neurons (N) operating in the subthreshold region and featuring sub-μW power consumption. The synapse circuit implements a four-quadrant multiplication function whereas the neuron circuit performs a *tanh*-like nonlinear activation function. Both circuits were designed by applying the translinear principle with FGTs being an integral part of the design. The FGTs have a dual role. They are used as compact nonvolatile analog weight storage solution after the training process is completed, providing up to 10-bits of precision for each stored weight. Additionally, they provide a reconfiguration mechanism for supporting various neural network topologies.

Apart from the core, three peripheral circuits provide support for the training and operation of the neural network. The differential transconductors (GM) accept differential voltage signals as inputs and convert them into differential currents, as required by the core. The current-to-voltage converter (ITOV) facilitates the reading of internal currents such as weights and neural network responses. Finally, the digitally-controlled current source (DCCS) generates high precision target currents (Iprog) for updating the values of weights during training. The DCCS is only used for programming. Therefore, in the proposed analog performance locking solution, it is not integrated on every die, as the end-user of the locked IC does not need to reprogram the neural network[2]. Instead, this is implemented on the instrumentation that will be used to program the fabricated devices prior to shipping them to the end-user.

For the purpose of locking the LNA performances, we used the multilayer perceptron (MLP) model. The MLP is a feed-forward neural network wherein each neuron receives connections only from inputs or previous layers. In this work a two-layer MLP neural network was trained to act as the lock. The first layer, which is called hidden layer, receives the inputs, i.e., the analog key values, as well as a bias. The number of hidden neurons reflects the learning capacity of the model. The second layer, which is called output layer, receives the outputs of the first layer and a bias, and produces the network outputs (in our case, the three bias voltages needed by

[2]In fact, to prevent the unauthorized end-user from doing so, a similar fuse-based solution to the one described in section III(B) for preventing observability of the neural network outputs, is applied to prevent retraining of the analog neural network once it is trained.

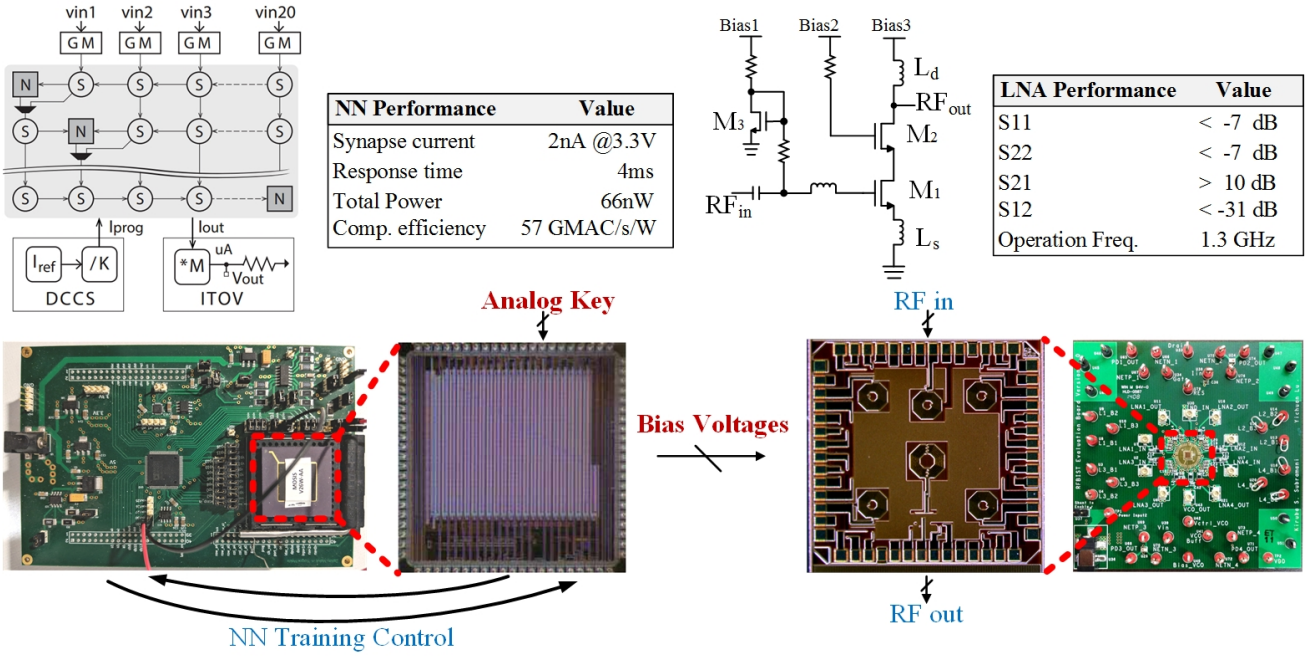

Fig. 3. Experimental Platform for Emulating the Proposed Analog Performance Locking Method

the LNA). The contribution of each connection to the output response is determined by the synaptic product of the local weight value and the corresponding input or hidden neuron output. The sum of the synaptic products is then applied to the neurons. In order to train the MLP neural network (i.e., to adjust its synaptic weights), a chip-in-the-loop training strategy along with a hardware-customized version of a resilient back propagation algorithm are employed [12].

Overall, our experimentation platform is an accurate emulation of the proposed analog performance locking solution.

V. RESULTS

Based on the aforementioned platform, we conducted several experiments seeking to elucidate three aspects of bias-based analog performance locking: (i) the inadequate security offered by using biases directly as analog keys, (ii) the effectiveness of the proposed method against both brute force and model approximation attacks, and (iii) the impact of bias obfuscation on the performances of the LNA.

A. Analog Performance Locking Through Biases

The effect of the three bias voltages on the fabricated LNA performance was studied by measuring the four S-parameters (S11, S12, S21, S22) for various combinations of these biases. The measured results are shown in Figure 4. In the plot of each of the four S-parameters, the three axis represent the three bias voltages and the color-coding represents the value of the S-parameter, as indicated in the corresponding color-bar. Based on these measurements, specification-compliant operation of the LNA is achieved only when (Bias1, Bias2, Bias3) = (1.35, 2.55, 2.25)V. For this specific combination of bias voltages (i.e., operating point) the performance of the LNA is optimized and corresponds to (S11, S21, S12, S22) = (-8, 11.2, -31.4, -7.5)dB. For any other combination of bias voltages, at least one S-parameter fails its specifications.

TABLE I
ABILITY TO LEARN RELATION BETWEEN BIASES AND PERFORMANCES

	Powells method	Gradient Minimization
Success Rate	100%	100%
Number of Samples	74.5	8

The exact values needed for three bias voltages in order for the LNA to be specification-compliant could be thought of as a key for locking its performances. However, such a solution would be very weak, due to the strong correlation between the LNA performances and the values of voltage biases. Indeed, an attacker can exploit this correlation and use model approximation algorithms to quickly deduce the needed bias voltage values for specification-compliant LNA operation. To demonstrate this point, we used two such algorithms, namely Powell's method [9] and Gradient Minimization [10], and we report the results in Table I. Starting from a randomly chosen set of initial values, both algorithms can successfully find the optimum bias voltages within a relatively small number of sampling points. Among the two algorithms, conjugate gradient minimization is much more effective, since it requires on average only 8 sampling points, confirming our conjecture that the direct use of bias voltages as keys is a very weak analog performance locking option.

B. Neural Network-Based Biasing

We now proceed to evaluate the effectiveness of the proposed method for analog performance obfuscation using neural network-based biasing. As explained in Section III, in order to act as an effective lock, the neural network must be trained to implement a function that follows the general form shown in Figure 1(a). In this experiment, the neural network was trained to approximate the impulse function shown in Figure 5.

This impulse function was chosen because, as we explain below, it minimizes the information available to an attacker

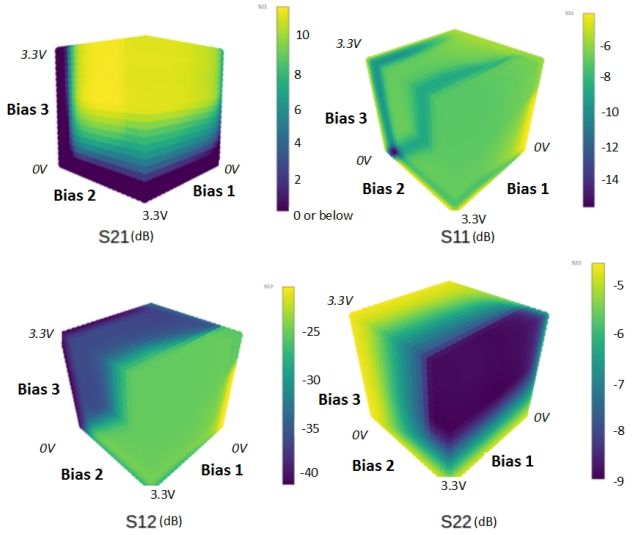

Fig. 4. Effect of Three Bias Voltages on LNA Performance

for staging model-based attacks. Specifically, when given the correct analog key at its inputs, the trained neural network generates the desired LNA bias voltages at its output. However, for any other input value, the trained neural network generates the exact same incorrect LNA bias voltages. This, in turn, results in non-compliant LNA performances, which do not vary as a function of the incorrect key values. As a result, model approximation algorithms, such as Powell′s method [9] and Gradient Minimization [10] cannot be applied.

We implemented a continuous approximation of the impulse function shown in Figure 5 using our analog neural network prototyping platform, as well as its equivalent software version using Matlab's neural network toolbox for comparison purposes. In each case, an MLP network with one hidden layer and, depending on the number of inputs, up to 10 hidden neurons was sufficient to implement the impulse function. To train the neural networks, we generated appropriate synthetic data-sets, which are dense around the correct key value and sparser as we move further away from this peak. The neural network has differential inputs whose range spans from -3.3V to 3.3V with a step-size of 2mV, corresponding to 3300 possible key options for each input. Evidently, the key space grows quickly as a function of the number of inputs, making brute-force attacks infeasible.

A perfectly trained neural network would only accept the exact values of the key as correct. However, due to training limitations, the software implementation may also accept a few additional values around the correct key. This phenomenon may be further exacerbated in the hardware implementation due to non-idealities. Therefore, in order to evaluate the quality of the lock implemented by the trained neural network, we use the metric described in Section III.C. Table II reports our results for software and hardware neural network implementations with up to three inputs.

For the software version of the neural network, using a 1-input key results in only 3 out of the 3300 possible values being accepted (i.e., generating 1.35V, 2.25V, and 2.55V for

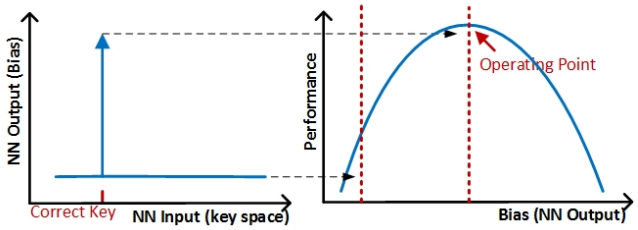

Fig. 5. Impulse Function for Bias Voltage Obfuscation

TABLE II
KEY GUESSING PROBABILITY METRIC FOR PROPOSED METHOD

Version	1 input	2 inputs	3 inputs
Software	$9.1\text{e}{-4} \approx 2^{-10}$	$8.3\text{e}{-7} \approx 2^{-20}$	$1.1\text{e}{-9} \approx 2^{-30}$
Hardware	$3.0\text{e}{-2} \approx 2^{-5}$	$1.1\text{e}{-3} \approx 2^{-9}$	$5.2\text{e}{-5} \approx 2^{-14}$

the three bias voltages at the output of the neural network), while all other key values result in a constant output (i.e., 0.8V, 0.8V, and 0.8V). In essence, this implies that the probability of guessing the correct key, as expressed by the metric of Section III(c), is approximately 2^{-10}, which makes it equivalent to a 10-bit digital key. When the number of neural network inputs increases to 2, 9 key values among the approximately 11M options are accepted, corresponding to a metric value of 2^{-20}, or the equivalent of a 20-bit digital key. This trend continues, with the metric value becoming 2^{-30}, or the equivalent of a 30-bit digital key for the trained 3-input software neural network. In the case of hardware, a similar exponential growth is observed in our experiments for 1-, 2-, and 3-input networks. However, due to hardware non-idealities and resolution limitations of the peripheral circuitry [8] that is used to train the neural network, the equivalent number of digital key bits is approximately half of the software version value.

In Figure 6, we project the observed trends for both the software and hardware neural network implementations as a function of the number of inputs. In the digital domain, common rule of thumb is that an 80-bit key space is virtually unbreakable through brute force attacks. As can be observed, in the case of software neural network implementation, 8 inputs are sufficient to reach this security level, while for the hardware version, 16 inputs are required. Nevertheless, we should point out that in analog circuits, due to settling and measurement complexity, evaluating effectiveness of a chosen bias voltage option takes much longer than in their digital counterparts. Therefore, it is likely that a much smaller number of key inputs will suffice to match the time required for a brute force attack on an 80-bit digital key.

C. Bias Obfuscation Impact on LNA Performances

To further demonstrate the effectiveness of neural network-based biasing in locking the performances of the LNA, Figure 7 reflects the value of the second bias voltage (i.e., Bias2) for the entire input space of a trained 2-input hardware neural network. As explained in the previous subsection and as shown in Table II, the percentage of points around the correct key value which is accepted by this version of the trained neural network (i.e., which produces Bias2=2.55V) is 2^{-9}. Similar

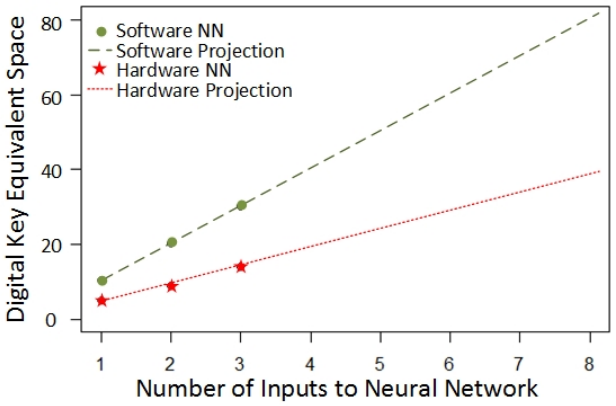

Fig. 6. Projection of Key Space as a Function of Number of Inputs

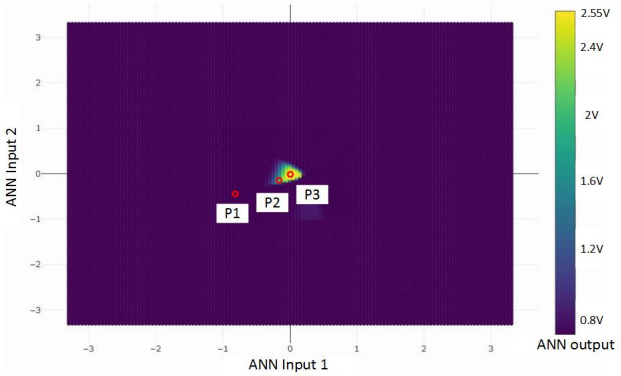

Fig. 7. Function Learned by 2-Input Hardware Analog Neural Network

figures can be plotted for the other two bias voltages. In Figure 7, we highlight three points (i.e., P1, P2, P3) in the key space in order to explain the impact of the proposed locking on the LNA performance, which is summarized in Table III. If Key P1 is applied, the neural network will generate three bias voltages of 0.8V, for which the LNA will fail its specifications, i.e., its performances will be successfully locked. the vast majority of points in the input key space belong to this category. If the correct key P3 is applied, the neural network will produce the desired bias voltages of 1.35V, 2.55V, and 2.25V, for which the S-parameters will be within their specifications and the LNA will be successfully unlocked. Only one point in the input key space belongs to this category. Ideally, in a perfectly trained neural network, these would be the only two categories of points. However, due to training non-idealities, there exists a third category which is represented by point P2. For a small number of points around P3, the neural network will generate bias voltages which will result in S-parameters that may still not meet the LNA specifications, but are sufficiently different than the S-parameters of P1. Such points can serve as clues regarding the direction in which a model approximation algorithm should search; therefore, we consider points such as P2 as successful keys for unlocking the LNA performances.

TABLE III
IMPACT OF PROPOSED LOCKING ON LNA PERFORMANCES

Points	Status	Biases (V)	Performance (dB)			
			S11	S21	S12	S22
P1	locked	0.8, 0.8, 0.8	-6.5	0.3	-34.9	-4.8
P2	in-between	1.05, 1.5, 1.35	-6.9	8.1	-33.9	-5.6
P3	unlocked	1.35, 2.55, 2.25	-8	11.2	-31.4	-7.5

VI. CONCLUSIONS

We presented a method for locking the performances of an analog IC and preventing its unauthorized use, by controlling its biasing through an analog neural network. The neural network is trained to generate the required bias voltages if and only if the correct analog key is provided at its inputs, while for any incorrect key value it generates incorrect biases for which the circuit fails its specifications. Since the neural network is programmed after fabrication, the key can be individualized per customer or even per device to further enhance security. To demonstrate the proposed analog performance locking method, we developed an experimentation platform which comprises two fabricated ICs, an LNA and an analog neural network prototyping chip. Experimental results confirmed the effectiveness of the proposed method against brute force and model approximation attacks with even a small number of analog key inputs. To our knowledge, this is the first method employing analog keys for the purpose of locking analog IC performances.

ACKNOWLEDGMENT

This research has been partially supported by National Science Foundation (NSF) and Semiconductor Research Corporation (SRC) through tasks 1527460 and 2625.001, respectively.

REFERENCES

[1] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 1069–1074.

[2] R. S. Chakraborty and S. Bhunia, "Hardware protection and authentication through netlist level obfuscation," in *Proceedings of Int. Conference on Computer-Aided Design*, 2008, pp. 674–677.

[3] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing ic piracy using reconfigurable logic barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, 2010.

[4] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 83–89.

[5] V. V. Rao and I. Savidis, "Protecting analog circuits with parameter biasing obfuscation," in *Test Symposium (LATS), 18th IEEE Latin American*, 2017, pp. 1–6.

[6] J. Wang, C. Shi, A. Sanabria-Borbon, E. Sánchez-Sinencio, and J. Hu, "Thwarting analog ic piracy via combinational locking," in *Test Conference (ITC), IEEE International*, 2017, pp. 1–10.

[7] Y. Lu, K. S. Subramani, H. Huang, N. Kupp, K. Huang, and Y. Makris, "A comparative study of one-shot statistical calibration methods for analog/RF ICs," in *in Test Conference (ITC), IEEE International*, 2015, pp. 1–10.

[8] D. Maliuk and Y. Makris, "An experimentation platform for on-chip integration of analog neural networks: A pathway to trusted and robust analog/RF ICs," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1721–1734, 2015.

[9] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.

[10] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*. NBS Washington, DC, 1952, vol. 49, no. 1.

[11] G. Volanis, D. Maliuk, Y. Lu, K. S. Subramani, A. Antonopoulos, and Y. Makris, "On-die learning-based self-calibration of analog/RF ICs," in *VLSI Test Symposium (VTS), IEEE 34th*, 2016, pp. 1–6.

[12] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *Neural Networks, IEEE International Conference on*, 1993, pp. 586–591.