Chaogate Parameter Optimization using Bayesian Optimization and Genetic Algorithm

Rabin Yu Acharya[1], Noeloikeau F. Charlot[2], Md Mahbub Alam[3], Fatemeh Ganji[4],
Daniel Gauthier[2], and Domenic Forte[1]
[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA
[2]Department of Physics, The Ohio State University, Columbus, OH, USA
[3]Technology Manufacturing Group, Intel Corporation, Chandler, AZ, USA
[4]Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA, USA
Email: rabin.acharya@ufl.edu, charlot.5@buckeyemail.osu.edu

***Abstract*—Chaotic circuits have found application in various research areas, including cryptography. However, more effort has to be made to achieve the properties required for such circuits when it comes to their circuit design. We identify and optimize for regions of chaos in a simple three-transistor system known as a chaogate. We use simulations to study the dynamical behavior of the system treated as a one-dimensional map, and then maximize its chaotic and cryptographic behavior using artificial intelligence. We propose several useful metrics for the chaogate, such as the maximum Lyapunov exponent, and measure these metrics over the transistor parameter space. Finally, we apply Bayesian optimization and Genetic Algorithm to identify various chaogate designs in different technology nodes, which we visualize, compare, and use to propose future research.**

***Index Terms*—Chaos, Chotic circuits, Chaogate, Cryptography, AI, Optimization.**

I. Introduction

The chaogate is a dynamical circuit that exhibits highly nonlinear behavior known as chaos [1]. Chaos is often characterized by an exponential but deterministic divergence of nearby trajectories in a dynamical system over time. This property has been leveraged in a variety of applications including cryptography and computation. Research into microelectronic chaotic primitives such as the chaogate is therefore essential to make advances in these fields.

In 1998, it was proposed to use chaotic systems to realize a new class of computing devices [2]. At the time, the focus was on proof-of-principle demonstrations of the capability of chaotic elements for universal computing by exploiting the sensitivity and pattern formation features. For example, the dynamical system proposed in Refs. [3] and [4] was among the first to use a single chaotic element as a reconfigurable logic gate. The chaos-based arithmetic logic unit (ALU) described in Ref. [5] was later used to construct a multi-input, multi-output logic block. Similarly, a gate using Chua's circuit was used to implement multiple logic gates [4]. In Ref. [6], a simple nonlinear circuit, so-called "chaogate," was claimed to implement an infinite number of functions. Other chaos sources described in the literature are the modified two-transistor resistor-capacitor phase-shift oscillator [7], modified Wien bridge oscillator [8], driven inductor-varactor resonator [9], and the autonomous circuit using operational amplifiers and linear time-invariant passive components described in [10].

Chaotic systems have also found use in security-related applications because of their inherent properties – sensitivity to initial conditions, unpredictability, deterministic nature (under known initial conditions), and random-like behavior. These properties are tailored to the requirements of various modules in cryptosystems, including compression, encryption, and modulation schemes [11]–[13]. In Refs. [12] and [11], chaos-based public-key cryptography and a chaotic-driver-based encryption scheme were proposed. Chaogates have also been considered as a promising low-cost solution to protect cryptosystems against non-invasive side-channel attacks [5], [14]–[16] because they exhibit a difficult-to-predict state-space of multiple implementations for various logical operations. In addition, chaotic systems for IC authentication, locking key exchange, etc. are considered to address the lack of (sub-)key and clock generation in analog chips [17].

In these previous works, a dynamical system is constructed in an *ad hoc* manner, and its inherent chaotic properties are not examined precisely. In particular, the parameters affecting the chaotic operation, and consequently, the security of the system embodying the chaotic circuit, are not clearly defined. They also fail to describe a clear methodology to design chaotic circuits based on chaos-related metrics and properties.

To begin to develop a design strategy, we first highlight that a chaotic system is a complex non-linear system that depends on several internal parameters. For example, the chaotic operation of the chaogate, composed of three MOSFET transistors [6], depends on each transistor's parameters and gate voltages. The three transistor sizes alone comprise of a large search space, which can be intimidating for a designer to explore. In that regard, several methods were proposed in the 2000's to design a circuit automatically such as in Refs. [18] and [19], which optimize the parameters of an analog IC using *artificial intelligence (AI)*. However, the use of these AI-based techniques stalled because it was not possible to develop a smooth accurate model of the transistor as a function of transistor parameters without expending substantial computational resources if at all. Recent advances in accurate MOSFET models for modern process technologies, AI applications, and availability of more powerful computational resources, however, are reducing such concerns. In light of these issues, our

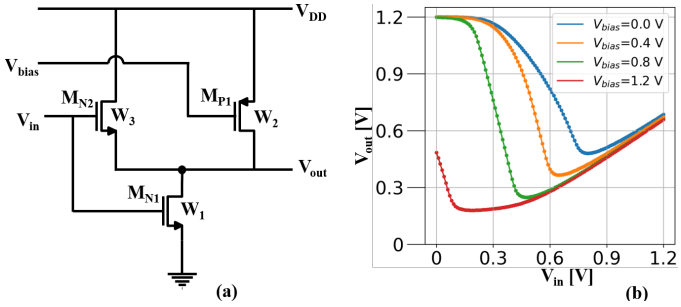

Figure 1: (a) A three-transistor (3T) chaogate; (b) V_{out} versus V_{in} for different V_{bias} and a specific set of widths.

main contributions in this paper are summarized as follows.

- Identify MOSFET parameters, such as transistor width and bias voltage, that affect chaotic behavior, especially, for the chaogate as an example of a chaotic circuit. In line with this, we develop a methodology to encode or represent the corresponding netlists of chaogate circuits in Python such that these parameters can be tuned easily.
- Formulate an objective function based on the chaos-related metrics, which is then used by state-of-the art AI optimization algorithms to obtain a chaotic circuit. We stress that the optimization problem underlying the design of chaotic circuits has not been well understood in the literature. Our results provide key insight into which type of optimization algorithms (*i.e.,* data-efficient and time-efficient) could be helpful in this regard.
- Simulate the generated netlists using multiple technology nodes and analyze their performances.

In the next section, we introduce the chaogate structure, operation, and theory. Section III discusses our proposed design methodology, including optimization approaches and parameters. In Section IV, we describe our simulation setup and results. We conclude with a summary and future work.

II. Preliminaries

Chaogate Design: In this paper, we use the chaogate design as implemented by Kia *et al.* [6]. This design, as shown in Fig. 1a, consists of three MOSFET transistors – 2 NMOS transistors (M_{N1} and M_{N2}) and one PMOS transistor (M_{P1}) – which are used to implement a nonlinear map to generate chaotic signals. The transistors M_{N1} and M_{N2} share the same input voltage V_{in}. The bias voltage V_{bias} (also called the control voltage in the literature) is connected to the gate of M_{P1}, which is used to change the response of the circuit. The resulting transfer characteristics curve of V_{out} versus V_{in} for different bias voltage is a V-shaped curve, which grows deeper with the increasing bias voltage as shown in Fig. 1b.

The V-shaped output curve has two regions – the left region with a negative slope and the right region with a positive slope whose behavior can be understood as follows. Consider an increasing input voltage V_{in}. When V_{in} is less than the threshold voltage of transistor M_{N2} ($V_{TH,M_{N2}}$), the output of the circuit is at a voltage level close to V_{DD} because both M_{N1} and M_{N2} are off. As V_{in} increases and is larger than $V_{TH,M_{N2}}$, M_{P1} and M_{N2} operate as an inverter and create the left region of the curve. M_{N1} is still off because the gate-source voltage $V_{GS,M_{N1}}$ is smaller than the $V_{TH,M_{N1}}$. When this voltage is greater than $V_{TH,M_{N1}}$, M_{N1} turns on to create the right region of the curve.

The core analog chaogate circuit shown in Fig. 1a is embedded in clocked circuitry to enable its operation (not shown) and described briefly here. The initial input to the chaogate circuit is set by the user and is directed to V_{in} using a switch. The subsequent value of V_{out}, after transient behavior has died out and the circuit settles to a steady-state behavior, is measured by a sample-and-hold circuit driven by a system clock. The switch is toggled so that it now directs V_{out} to V_{in}, and this process continues for a time controlled by the user.

Chaogate Theory: The dynamics of the chaogate can be described by

$$V_{n+1} = f(V_n; \boldsymbol{\alpha}), \quad n = 0, 1, 2, 3, ..., N \tag{1}$$

where N is the total number of iterations, V_{n+1} is the output voltage at the n^{th} iteration and V_n is the corresponding input voltage, f is the transformation given by the chaogate, and $\boldsymbol{\alpha}$ represents some specified list of chaogate parameters. This is known as a one-dimensional discrete-time ***map***, and, when iterated at fixed $\boldsymbol{\alpha}$, produce sequences

$$\{V_n\}|_{\boldsymbol{\alpha}} = \{V_0, f(V_0; \boldsymbol{\alpha}) = V_1, f(V_1; \boldsymbol{\alpha}) = V_2, ...\}, \tag{2}$$

with an example illustrated in Fig. 4. Even slight changes in $\boldsymbol{\alpha}$ around some critical point may be enough for the chaogate to transition into and out of chaos. This is why a careful study of the parameter space for chaogate operation is required, something which has yet to appear in the literature.

III. Proposed Design Methodology

The chaogate design procedure involves finding $f(V_{in}; V_{bias})$ ($\boldsymbol{\alpha} = \boldsymbol{W}, V_{bias}$) for a particular chaogate netlist, where $\boldsymbol{W} = (W_1, W_2, W_3)$ represent the widths of transistors M_{N1}, M_{P1}, and M_{N2}, respectively. This source of chaos, as studied by Dudek *et al.* [20] and Kia *et al.* [6], offers robustness against process variation, which can affect the slope of the transfer characteristics shown in Fig. 1b.

Based on these considertations, our general procedure for analyzing and designing a chaogate system is as follows.

1) For these parameters, V_{out} is recorded for a discrete sequence of V_{in}. This recording is done for different values of V_{bias} to obtain $f(V_{in}; V_{bias})$ used in the map given by Eq. 1.
2) This two-dimensional set of data is then used to calculate smooth spline interpolations of the chaogate map, which is used to iterate the map for values V_{in} over the domain used the generate the discrete set of points. Then, a transient sequence is constructed, where the transient is the first T elements such that the sequence diverges from its initial value if its chaotic. The number of iterations N and the transient length T is predetermined.
3) The Lyapunov exponent is then calculated using

$$\lambda(\boldsymbol{\alpha}) = (1/N) \sum_{n=T}^{n=N+T} |\ln(f'(V_n; \boldsymbol{\alpha}))|. \tag{3}$$

where f' is the derivative of the corresponding spline. The Lyapunov exponent is a logarithmic measure of the rate at which the sequence diverges, and a positive Lyapunov exponent ($\lambda > 0$) guarantees chaos for one-dimensional maps [21]. We use this to identify regions of V_{bias} for which the system is chaotic.

4) An objective function based on the Lyapunov exponent O_λ defined as $\max_\alpha(\lambda)$ or λ_m is devised. The optimization algorithms discussed in Secs. III-B and III-C are used to satisfy this objective function and return a set of widths and bias voltages such that the system is chaotic.

A. Parameters Affecting Chaogate Operation

As discussed in the previous section, the threshold voltage of the transistors play a huge role in determining the behavior of the circuit and thus the chaotic behavior. The other parameter which we have not yet discussed but is equally important is the ratio (W/L). For an ideal N-type MOSFET or NMOS transistor (ignoring channel-length modulation), the current through the transistor is given by [22]

$$I_{DS} = \begin{cases} 0, & V_{GS} < V_{TH}, \\ \mu_n C_{ox} \left(\frac{W}{L}\right) (V_{GS} - V_{TH}) V_{DS} & V_{DS} < V_{DSAT}, \\ \frac{1}{2} \mu_n C_{ox} \left(\frac{W}{L}\right) (V_{GS} - V_{TH})^2 & V_{DS} > V_{DSAT}, \end{cases} \quad (4)$$

where μ_n is the electron mobility, C_{ox} is the gate oxide capacitance, V_{TH} is the threshold voltage of the transistor, and V_{DSAT} is the voltage level at which the current saturates. Similar expressions are available for PMOS transistors. From Eq. (4), we see that, for a specific voltage instance of V_{GS} (voltage across gate-source terminal of the transistor) and V_{DS} (voltage across drain-source terminal of the transistor), I_{DS} is directly proportional to (W/L). This then affects the transfer characteristics of the circuit, which can dramatically affect the chaotic behavior of the circuit as well. Thus, the transistor sizes, specified by (W/L), must be chosen meticulously to obtain the desired chaotic operation. We choose L to be the minimum length allowed by the process technology node. The widths $\boldsymbol{W}$ and V_{bias} are chosen using the optimization algorithms - Bayesian optimization (BO) and Genetic Algorithm (GA).

Brief comparison of the optimization algorithms: BO is considered one of the most data-efficient frameworks for optimization tasks, but it can be computationally expensive with complexity $O(n^3)$ where n is the number of evaluations of candidate solutions. GA, on the other hand, is data inefficient but computationally much less intensive than BO. The search heuristics used by evolutionary algorithms in GA take constant time for generating candidate solutions [23]. As such, one has clear advantage over the other depending on the task at hand. However, for tasks that fall in the middle of the two (*i.e.*, with a moderate evaluation cost), choosing one over the other becomes difficult. As the optimization problem that we deal with has not been thoroughly studied before, choosing between BO and GA seems non-trivial. Therefore, we have applied both algorithms separately and presented the results.

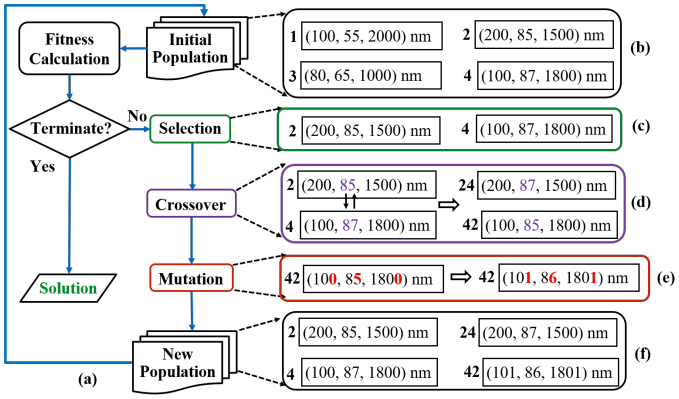

Figure 2: (a) GA Flowchart with example showing (b) population initialization; (c) selection of fit members from the population; (d) crossover between fit members to create new members; (e) mutation of genes in certain members; and (f) the new population.

B. Bayesian Optimization

Bayesian optimization [24] is a statistical approach for finding the maximum of an objective function f that is costly to evaluate over the full range of its argument(s) $x \in X$. The technique is based upon Bayesian statistics, in which conditional probability distributions are continuously updated as new data is acquired. In practice, Bayesian optimization techniques amount to algorithms that sample the space X to maximize the conditional probability that a particular point $x^* = \text{argmax}_{x \in X}[f(x)]$ is the location of the function maximum, given all previous observations.

To this end, various sampling algorithms have been devised, though the most common are Gaussian processes, which draw samples assuming independence among the arguments. Similarly, one may provide a prior probability distribution encoding the likelihood that the function maximum lies at each point in the space. The most common prior is a uniform prior, expressing an initial assumption that the function maximum is equally likely to occur anywhere.

Here, we consider a uniform prior and a Gaussian process as the sampling algorithm of our Bayesian optimizer (BO). Details on the parameter space, objective function, and implementation are given in Sec. IV.

C. Optimization using Genetic Algorithm

Genetic Algorithm (GA) is an evolutionary-based optimization technique inspired by Darwin's theory of natural selection. The basic idea in GA is to find the fittest individual or the best solution over a specific search space using three evolutionary operators - selection, crossover, and mutation - that are applied to a population of *chromosomes* as shown in Fig. 2a. Chromosomes are individuals that represent potential solutions to a problem (*e.g.*, 1 through 4 represents different chromosomes in Fig. 2b). A chromosome is typically represented as a string of binary bits or real numbers called *genes*. During the start of the algorithm, a large population of these chromosomes is created at random. Each chromosome is assigned a metric value based on the *fitness function*, which determines how good the chromosome is at solving the problem. If it solves

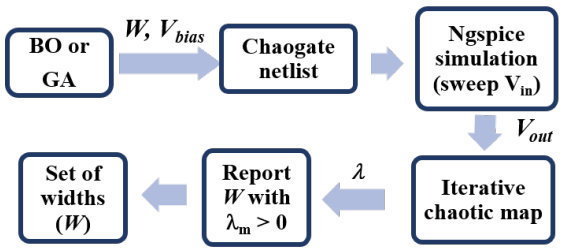

Figure 3: Simulation setup showing the procedure used to obtain a chaotic set of widths.

the problem, then the algorithm halts. Otherwise, depending on the fitness values, only a few chromosomes are selected as denoted by the step *selection* (see Fig. 2c). The next steps involve creating new chromosomes, which is accomplished by swapping genes of two fit chromosomes in a process called *crossover*, as shown in Fig. 2d, and by mutating certain genes of the chromosomes in a process called *mutation*, as shown in Fig. 2e. This concludes one *run* or *generation* of the GA algorithm and this evolution process continues until the fittest chromosome is created or a stopping criterion is met [25]. In a particular GA, only a fraction of the chromosomes are replaced, and the selection process is biased towards the fitter individuals.

GA optimization for chaogate design: In the context of chaogate design, our fitness function is the Lyapunov exponent (Eq. (3)) and the fittest chromosome is the one with the highest Lyapunov exponent. For this analysis, a stopping criteria can be provided such as $\lambda = k$, where k is a positive real number, or the algorithm is allowed to run for fixed number of iterations or generations after which the values of $\boldsymbol{W}$ with the maximum λ are selected for the chaogate.

IV. Simulation Results and Analysis

A. Simulation Setup

We simulate the netlist shown in Fig. 1a in multiple commercial technology nodes with 65 nm assumed when not specified. A DC sweep of V_{out} versus V_{in} with $V_{bias} \in [0, 1.2]$ V and V_{DD}=1.2 V is performed to obtain the transfer characteristics like those in Fig. 1b. As discussed earlier, we optimize $\boldsymbol{W}$. The creation of netlist, the optimization of transistor parameters, and the corresponding simulations (with new width values) are entirely done within the Python environment as shown in Fig. 3. We use PySPICE [26] v. 1.4.3, which is an open-source module that allows the user to simulate and manipulate SPICE netlists within Python by interfacing to the Ngspice v. 33 simulator.

B. Simulation Procedure

In the simulations, we use the procedures discussed in Sec. III for generating $f(V)$. At intermediate steps, a convenient method for visualizing the dynamics is through a *cobweb diagram* [21], which overlays the trajectories on the transfer function with an example shown in Fig. 4. We iterate the map to produce a cobweb diagram that describes the behavior of the system - a closed polygon indicates a repetitive sequence

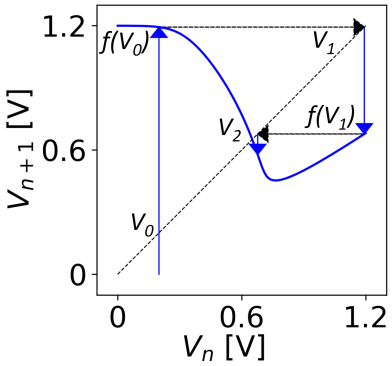

Figure 4: Cobweb diagram showing the first two iterations $\{V_n\} = \{V_0, V_1, V_2\}$ of the chaogate map $V_{n+1} = f(V_n)$ starting at $V_0 = 0.2$ V for $V_{bias} = 0.1$ V.

of fixed values (Fig. 5a) while a filled region of non-repetitive values suggests chaos (Fig. 5b). This behavior of the system can also be examined using a bifurcation diagram (Fig. 5c) which shows the values visited by the system as the bifurcation parameter (V_{bias} in our case) changes. From Fig. 5c, we observe regions of chaos and windows of periodic behaviour of the chaogate circuit. Lastly, we use BO and GA to suggest desirable $\boldsymbol{W}$ for chaogate design.

For each sequence $\{V_n\}|_{\boldsymbol{\alpha}}$ (Eq. (2), Fig. 4), we find that $N = 2,500$ with $T = 500$ is enough to obtain representative sequences starting with an arbitrarily chosen $V_0 = 0.45$ V. We fix these quantities for the remainder of this simulation. Next, we calculate the Lyapunov exponent using Eq. (3). We use $\lambda > 0$ to identify regions of V_{bias} for different $\boldsymbol{W}$ such that the system is chaotic (Fig. 5, 6). Similarly, we calculate the Shannon entropy (H) of the long-term behavior of the probability density function of each chaogate iteration sequence using

$$H(\boldsymbol{\alpha}) = -\sum_{n=T}^{n=N+T} p_n(\boldsymbol{\alpha}) \log_2(p_n(\boldsymbol{\alpha})), \tag{5}$$

where $p_n(\boldsymbol{\alpha})$ is the frequency of observation of element V_n from the iterated map sequence $\{V_n\}|_{\boldsymbol{\alpha}}$.

C. Optimization Results and Analysis

As discussed in Sec. III, the optimization cost corresponding to the design of the chaogate has not been determined previously. Here, we consider both data- and time-efficient types, namely, BO and GA to optimize the function $O_\lambda = \max_\alpha(\lambda)$, which explores the width space to return a specific set $\boldsymbol{W}$. To allow a fair comparison, the same number of iterations is considered for the BO and GA. Moreover, the range of parameters searched to perform the optimization is also fixed. Specifically, for the BO, we use the Python package ***skopt*** and the function ***gp_mimimize*** with default parameters. For GA, we devise a simple evolutionary algorithm as described in Sec. III-C.

Results with objective of λ_m: We run the BO and the GA algorithms for 100 generations and the corresponding results are shown in Figs. 6b and 6c. We also compare these results

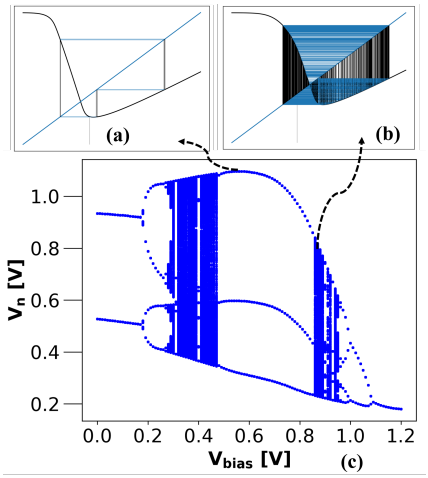

Figure 5: Cobweb plots of two regions showing (a) non-chaotic and (b) chaotic system behavior as V_{bias} is changed for fixed $\boldsymbol{W} = (120, 120, 2000)$ nm. (c) Bifurcation diagram, which shows the set of unique chaogate iteration points $\{V_n\}$ on the vertical for each bias voltage on the horizontal.

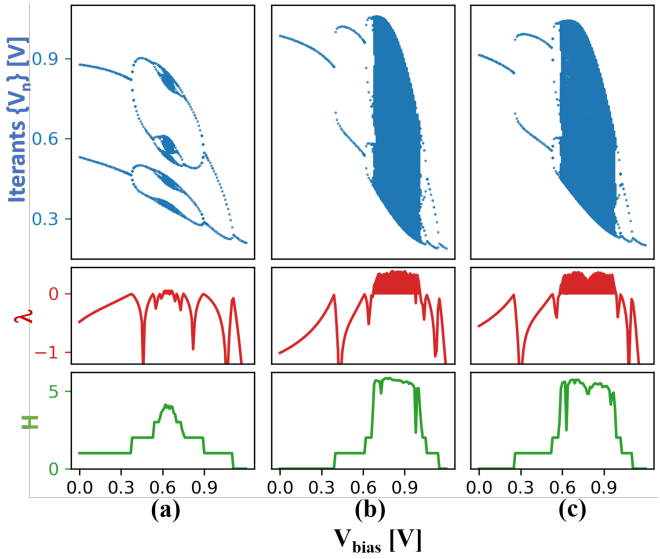

Figure 6: Dynamics of the chaogate (a) for $\boldsymbol{W} = (480, 480, 4800)$ nm and $\boldsymbol{L} = 60$ nm over the full range of V_{bias} [15]. (Blue) Bifurcation diagram showing set of map iterations $\{V_n\}$. (Red) Lyapunov exponent; shaded red regions are where $\lambda > 0$ and the system is chaotic. (Green) Entropy of the map iterations in bits, calculated by binning $\{V_n\}$ in 0.01 V increments. (b) with $\lambda_m = 0.40$ at $\boldsymbol{W} = (65, 247, 958)$ nm obtained using BO. (c) with $\lambda_m = 0.37$ at $\boldsymbol{W} = (66, 177, 2052)$ nm obtained using GA.

with the state-of-the-art chaogate circuit specified in Ref. [15] and shown in Fig. 6a. Both optimization algorithms are run to obtain a set of widths that achieve the largest Lyapunov exponent λ_m. The results obtained from these optimization algorithms look similar in that the change in H (the green steps) and the sign of λ (blue spikes) at each bifurcation are approaching the maximal density chaotic region around 0.8 V. Nevertheless, for the same number of iterations and range of

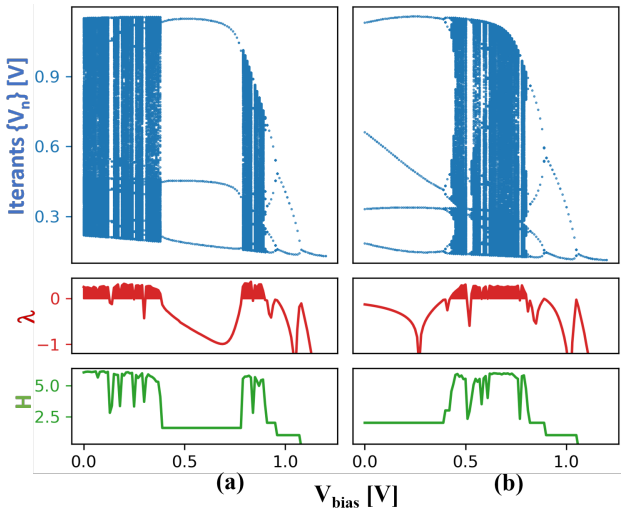

Figure 7: Dynamics of chaogate with the maximum entropy (a) $H_m = 6.18$ bits at $\boldsymbol{W} = (331, 65, 950)$ nm obtained using BO. Note the two bands of chaotic behavior, one of which indicates chaos even at $V_{bias} = 0$. Both regions are punctuated by periodicity characterized by drops in H and λ. (b) $H_m = 5.89$ bits at $\boldsymbol{W} = (1320, 87, 2060)$ nm obtained using GA.

parameters searched by the BO and GA, the GA converges to a result, where the λ's are positive over a wider range. In comparison to the chaotic system described in Ref. [15], the optimization algorithms clearly find circuits that show larger range of chaos over V_{bias}. As shown in Fig. 6a, this system has a very narrow chaotic region (the shaded red region) at around $V_{bias} = 0.6$ V. The wider chaotic region can also prove to be effective against process variation and noise, which will be the subject of our future work.

Results with objective of H_m: Now, if we change our objective from λ_m to the maximum value of the entropy H_m, the structure and position of the chaotic regions shift, giving us the result as shown in Fig. 7. The chaogate designer can certainly choose to have both λ and H maximized by providing a multi-objective function (λ_m, H_m) to the optimization algorithms. The maximum-entropy region shown in both Figs. 7a and 7b demonstrates a broad band of chaos at low bias voltages specifically Fig. 7a which is chaotic even at 0 V. This may be desirable for extremely low power cryptographic primitives. However, there exist small bands of periodicity in this region, which may interrupt cryptography.

In contrast, optimizing for λ, as shown in Figs. 6b and 6c, demonstrates the highest density chaotic region of the three (the red shaded region is larger compared to Fig. 6a). The set of states here may have applications in primitives prioritizing cryptography at the expense of power consumption. Additionally, the chaogates in these regions may have near maximal entropy. The 0.01 V bin space provides a total of $\log_2(120) = 6.91$ bits for our estimate. We see in each plot that $H_m \sim 6$ bits, naïvely suggesting an entropy density around $6/6.91 \approx 90\%$. More advanced entropy measures, such as the Kolmogorov-Sinai entropy, may yield more accurate results by taking into account information revealed by the ordering of $\{V_n\}$. This is the subject of future work.

TABLE I: Comparison of width values $\boldsymbol{W} = (W_1, W_2, W_3)$ obtained for different technology nodes, namely a commercial 65 nm and PTM 45 nm, 90 nm, and 130 nm. The widths are optimized to obtain maximum Lyapunov exponent using Bayesian Optimization (BO) and Genetic Algorithm (GA). The resulting H_m is also reported.

	BO			GA		
Node [nm]	$\boldsymbol{W}$ [nm]	λ_m	H_m	$\boldsymbol{W}$ [nm]	λ_m	H_m
45	(86, 47, 992)	0.48	5.39	(132, 67, 1983)	0.5	5.53
65	(65, 247, 958)	0.4	5.37	(66, 177, 2052)	0.37	5.41
90	(162, 149, 2032)	0.35	5.96	(114, 99, 2035)	0.35	5.93
130	(201, 236, 1882)	0.38	6.03	(207, 138, 2045)	0.34	6.05

D. Demonstration on Different Technology Nodes

Our optimization approach is flexible and can be applied to other technology nodes, chaotic circuits, and other analog circuits. Here, we use it to port chaotic behavior of the chaogate to different technology nodes, thus relieving the designer's burden. As seen in Table I, the optimizer returns a set of widths for different transistor technologies and it can be optimized to obtain either λ_m or H_m.

The key message that the results in Table I convey is that both the BO and GA could converge to (almost) similar widths regardless of the transistors technology. Nevertheless, to decide which algorithm should be chosen to perform the optimization task, time complexity can play a crucial role. To address this, we measure the time that the algorithms take to optimize the circuit 100 times for the commercial 65 nm model. The BO takes 157.3 s, while the GO finishes the task in 143.7 s. Similar trends have been observed for other transistor models. Note that although the difference between the time complexity of the BO and GA is not substantial in our scenario, this difference can affect the efficiency of the design process for larger or more complex analog circuits.

V. Conclusion and Future Work

AI is a promising approach for co-optimization of circuit security and performance. Here, we present a clear framework for designing chaotic circuits and explore the optimization problem underlying such a task in terms of its computational cost. As this cost has not been discussed in the literature, we have applied two well-known optimization methods, namely BO and GA borrowed from AI, which are less time-efficient and less data-efficient, respectively. In the current literature, it has been suggested to combine the two [23], which will be a subject of our future work. Moreover, future work includes tuning the hyperparameters of the optimization routine, testing other objective functions and constraints (including traditional circuit metrics such as area and power), performing a noise sensitivity analysis, and completely exploring the transistor-width parameter space to identify different trends. Additionally, the results obtained here will be used to fabricate chaogates in silicon for further experimental research.

References

[1] W. L. Ditto, A. Miliotis, K. Murali, S. Sinha, and M. L. Spano, "Chaogates: Morphing logic gates that exploit dynamical patterns," *Chaos: An Interdisciplinary J. of Nonlinear Science*, vol. 20, no. 3, p. 037107, 2010.

[2] S. Sinha and W. L. Ditto, "Dynamics based computation," *Phys. Rev. Lett.*, vol. 81, no. 10, p. 2156, 1998.

[3] T. Munakata, S. Sinha, and W. L. Ditto, "Chaos computing: implementation of fundamental logical gates by chaotic elements," *Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 11, pp. 1629–1633, 2002.

[4] S. Sinha and W. L. Ditto, "Computing with distributed chaos," *Phys. Rev. E*, vol. 60, no. 1, p. 363, 1999.

[5] G. S. Rose, "A chaos-based arithmetic logic unit and implications for obfuscation," in *Computer Society Annual Symp. on VLSI*, pp. 54–58, IEEE, 2014.

[6] B. Kia, J. F. Lindner, and W. L. Ditto, "A simple nonlinear circuit contains an infinite number of functions," *Trans. on Circuits and Systems II: Express Briefs*, vol. 63, no. 10, pp. 944–948, 2016.

[7] L. Keuninckx, G. Van der Sande, and J. Danckaert, "Simple two-transistor single-supply resistor–capacitor chaotic oscillator," *Trans. on Circuits and Systems II: Express Briefs*, vol. 62, no. 9, pp. 891–895, 2015.

[8] A. Namajunas and A. Tamasevicius, "Modified Wien-bridge oscillator for chaos," *Electronics Lett.*, vol. 31, no. 5, pp. 335–336, 1995.

[9] P. S. Linsay, "Period doubling and chaotic behavior in a driven anharmonic oscillator," *Phys. Rev. Lett.*, vol. 47, no. 19, p. 1349, 1981.

[10] J. R. Piper and J. C. Sprott, "Simple autonomous chaotic circuits," *Trans. on Circuits and Systems II: Express Briefs*, vol. 57, no. 9, pp. 730–734, 2010.

[11] L. Keuninckx, M. C. Soriano, I. Fischer, C. R. Mirasso, R. M. Nguimdo, and G. Van der Sande, "Encryption key distribution via chaos synchronization," *Scientific Reports*, vol. 7, p. 43428, 2017.

[12] I. Mishkovski and L. Kocarev, *"Chaos-Based Public-Key Cryptography"*, pp. 27–65. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[13] T. Stojanovski, J. Pihl, and L. Kocarev, "Chaos-based random number generators. Part II: practical realization," *Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 3, pp. 382–385, 2001.

[14] J. Bohl, L.-K. Yan, and G. S. Rose, "A two-dimensional chaotic logic gate for improved computer security," in *58th Intrl. Midwest Symp. on Circuits and Systems*, pp. 1–4, IEEE, 2015.

[15] M. B. Majumder, M. S. Hasan, M. Uddin, and G. S. Rose, "Chaos computing for mitigating side channel attack," in *Intrl. Symp. on Hardware Oriented Security and Trust*, pp. 143–146, IEEE, 2018.

[16] M. B. Majumder, M. S. Hasan, A. Shanta, M. Uddin, and G. Rose, "Design for Eliminating Operation Specific Power Signatures from Digital Logic," in *Proc. of the Great Lakes Symp. on VLSI*, p. 111–116, ACM, 2019.

[17] M. M. Alam, S. Chowdhury, B. Park, D. Munzer, N. Maghari, M. Tehranipoor, and D. Forte, "Challenges and opportunities in analog and mixed signal (ams) integrated circuit (ic) security," *J. of Hardware and Systems Security*, vol. 2, no. 1, pp. 15–32, 2018.

[18] M. Fakhfakh, Y. Cooren, A. Sallem, M. Loulou, and P. Siarry, "Analog circuit design optimization through the particle swarm optimization technique," *Analog Integrated Circuits and Signal Processing*, vol. 63, no. 1, pp. 71–82, 2010.

[19] M. Taherzadeh-Sani, R. Lotfi, H. Zare-Hoseini, and O. Shoaei, "Design optimization of analog integrated circuits using simulation-based genetic algorithm," in *Intl. Symp. on Signals, Circuits and Systems*, vol. 1, pp. 73–76, IEEE, 2003.

[20] P. Dudek and V. Juncu, "Compact discrete-time chaos generator circuit," *Electronics Lett.*, vol. 39, no. 20, pp. 1431–1432, 2003.

[21] H. G. Schuster and W. Just, *Deterministic chaos: an introduction*. John Wiley & Sons, 2006.

[22] T. Sakurai and A. R. Newton, "A simple MOSFET model for circuit analysis," *IEEE transactions on Electron Devices*, vol. 38, no. 4, pp. 887–894, 1991.

[23] G. Lan, J. M. Tomczak, D. M. Roijers, and A. Eiben, "Time Efficiency in Optimization with a Bayesian-Evolutionary Algorithm," *arXiv preprint arXiv:2005.04166*, 2020.

[24] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[25] R. Y. Acharya, S. Chowdhury, F. Ganji, and D. Forte, "Attack of the Genes: Finding Keys and Parameters of Locked Analog ICs Using Genetic Algorithm," in *Intrl. Symp. on Hardware Oriented Security and Trust*, pp. 284–294, IEEE, 2020.

[26] F. Salvaire, "PySpice." https://pyspice.fabrice-salvaire.fr, 2020.