Multiplex-free physical reservoir computing with an adaptive oscillator

Md Raf E Ul Shougat [1], XiaoFu Li [2], and Edmon Perkins [2,*]

[1]*Department of Mechanical and Aerospace Engineering, North Carolina State University, Raleigh, North Carolina 27695, USA*

[2]*LAB2701: Nonlinear Dynamics Laboratory, Atwood, Oklahoma 74827, USA*

(Received 18 July 2023; revised 21 November 2023; accepted 8 January 2024; published 7 February 2024)

Nonlinear oscillators can often be used as physical reservoir computers, in which the oscillator's dynamics simultaneously performs computation and stores information. Typically, the dynamic states are multiplexed in time, and then machine learning is used to unlock this stored information into a usable form. This time multiplexing is used to create virtual nodes, which are often necessary to capture enough information to perform different tasks, but this multiplexing procedure requires a relatively high sampling rate. Adaptive oscillators, which are a subset of nonlinear oscillators, have plastic states that learn and store information through their dynamics in a human readable form, without the need for machine learning. Highlighting this ability, adaptive oscillators have been used as analog frequency analyzers, robotic controllers, and energy harvesters. Here, adaptive oscillators are considered as a physical reservoir computer without the cumbersome time multiplexing procedure. With this multiplex-free physical reservoir computer architecture, the fundamental logic gates can be simultaneously calculated through dynamics without modifying the base oscillator.

DOI: 10.1103/PhysRevE.109.024203

I. INTRODUCTION

Nonphysical reservoir computers are constructed from recurrent neural networks with some hidden layers (i.e., the *reservoir*) that have random, untrained weights. The output of the reservoir is trained with a simple method, such as a ridge regression. A physical reservoir computer (PRC) effectively replaces the software recurrent neural network with a physical system. Said another way, input information is sent to a physical system as external force(s), the physical reservoir transforms and stores this information as a dynamic response, and the dynamic response is trained (e.g., with ridge regression) to produce a desired output. A schematic of this process for the adaptive oscillator is shown in Fig. 1.

In physical reservoir computing, the dynamic response of a nonlinear system is used as a computational resource by applying machine learning techniques [1–6]. Reservoir computing came from *liquid state machines* [7] and *echo state networks* [8]. Since PRCs do not have static information storage, they are different from Turing machines. Many different time-dependent systems have been explored as physical reservoir computers. These include the Hopf oscillator [9–11], van der Pol oscillator [12], memristors [13], a Duffing array [14], quantum reservoir networks [15], superparamagnetic tunnel junctions [16], an array of linear oscillators [17], spintronics [18], lasers in special mediums [19], microelectromechanical systems [20], a shape memory alloy actuator [21], and the nonlinear response of materials [22].

On the other hand, adaptive oscillators are a subset of nonlinear oscillators that can learn and store information in dynamic plastic states. For instance, the Hopf adaptive frequency oscillator has two states that correspond to the nonadaptive Hopf oscillator, while the remaining state can learn and store an external forcing frequency [23,24]. This can be extended to include states for amplitude [25–27] or more complex waveforms [28]. Since linear vibratory energy harvesters employ resonance to boost performance [29] and nonlinear harvesters use nonlinearity to increase bandwidth [30,31], adaptive oscillators may be ideal vibratory energy harvesters since they can track resonance (e.g., the pendulum adaptive frequency oscillator [32]). Interestingly, adaptive oscillators can also exhibit chaotic motion for certain parameter combinations [33].

Oscillator-based physical reservoir computers typically employ time multiplexing to create virtual nodes [34]. For vibrating systems, this is a seemingly necessary step to capture the necessary information from the system, as computing and memory storage are both provided by the oscillator itself. This is cumbersome from a practical perspective, as it requires a faster sampling rate than the clock frequency. For time multiplexing, a piece of information (e.g., a logic bit, a single sample from an audio recording, or a pixel, etc.) is sent to the oscillator, and the oscillator's response is sampled n times. These n samples of the oscillator are considered to be *virtual node*s, as they are sampled in time from a single state of the oscillator. Conversely, a sensor array might instead be sampled when the array is subjected to a force; thus, for spatial multiplexing, the nodes are physical. Effectively, both time and spatial multiplexing are employed to project a single piece of data to a vector in a larger space. For a physical reservoir computer composed of an adaptive oscillator, time multiplexing can be completely avoided. As only a single node is collected for each pseudoperiod, the nodes considered in this paper are physical nodes, even though it is an oscillator. The adaptive oscillator's response at a single instance per

*edmon@lab2701.com

clock cycle directly provides the calculation of different logic gates. Interestingly, this approach creates a reprogrammable logic gate, without the necessity of modifying the adaptive oscillator itself.

As a relevant comparison to the present paper, nonlinear oscillators have also been used for reprogrammable logic gates, in which the dynamics of the oscillator are modified to correspond as different gates. For instance, a bistable oscillator's asymmetry can be tuned to produce different logic gates [35], lattices of coupled chaotic maps set a critical value to function as different gates [36], and a Duffing oscillator can be used as a reprogrammable logic gate by incorporating a feedback controller [37,38]. The present adaptive oscillator PRC utilizes the complex dynamic response of the oscillator to avoid modifying the oscillator itself (i.e., no feedback or parameter tuning is necessary). At the same time, machine learning is used to reprogram the nonmultiplexed response to correspond to different logic gates using a single sample per clock cycle. Since this is done without modifying the base oscillator's parameters, all of the logic gates can be simultaneously calculated in parallel by the physical reservoir computer.

II. ADAPTIVE OSCILLATOR PHYSICAL RESERVOIR COMPUTER

The adaptive oscillator considered in this paper is based on the Hopf oscillator, whose equations of motion are given below:

$$\begin{aligned} \dot{x} &= [\mu - (x^2 + y^2)]x - \omega_0 y, \\ \dot{y} &= [\mu - (x^2 + y^2)]y + \omega_0 x. \end{aligned} \tag{1}$$

Here, μ is a parameter that affects the limit cycle radius, and ω_0 is the static resonance frequency of the Hopf oscillator. It should be noted that other adaptive oscillators can be constructed with different base oscillators, and these other adaptive oscillators should exhibit a similar performance as physical reservoir computers. By concatenating an additional state onto Eq. (1), the Hopf-based adaptive oscillator is constructed that can learn and store frequency information [25–27]:

$$\begin{aligned} \dot{x} &= [\mu - (x^2 + y^2)]x - \omega y + k_x f(t), \\ \dot{y} &= [\mu - (x^2 + y^2)]y + \omega x, \\ \dot{\omega} &= -k_\omega f(t) y. \end{aligned} \tag{2}$$

When this adaptive oscillator is forced with a single sinusoid, such as $a \sin(\Omega t)$, the ω state will converge to Ω. k_x and k_ω are constants that affect the learning rate of the adaptive oscillator. In Eq. (2), $f(t)$ is a time-varying external force, which encodes information that is sent to the adaptive oscillator. For the logic tasks considered here, "True" is encoded as Ω_{TRUE} and "False" is encoded as Ω_{FALSE}, and the external forcing function is defined simply as

$$f(t) = \sin(\Omega_i t). \tag{3}$$

The "True" or "False" values are chosen at random with equal probability, which are then encoded as Ω_i. This frequency value is held constant for a pseudoperiod T_p that is significantly longer than the period of Ω_{TRUE} and Ω_{FALSE}. The pseudoperiod can be considered as the clock frequency, such that a logic gate is calculated once for each pseudoperiod. As no time multiplexing is used, the nodes for each clock are simply a bias value (e.g., 1) and the ω state value. The ω values are taken just after each clock cycle to capture the adaptive oscillator in a transient state. Thus, the nodes are $[1, \omega(nT_p + \delta)]$, where $n \in \mathbb{Z}^+$ and δ is a small constant. An example of the time history of the adaptive oscillator PRC is shown in Fig. 2.

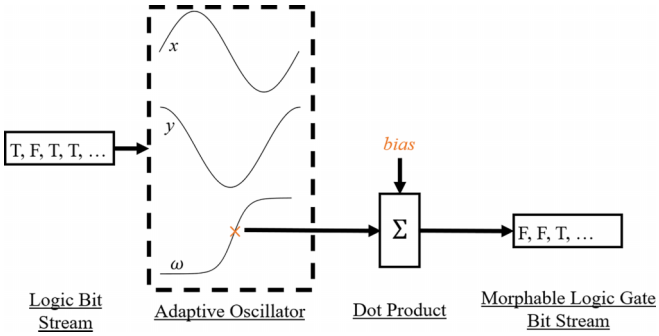

FIG. 1. Schematic of the adaptive oscillator physical reservoir computer considered here. A logic bit stream is sent to the adaptive oscillator, and only a single node is collected from the ω state for each clock cycle. Thus, the oscillator must "remember" the previous bit to perform the logic gate correctly. The Σ denotes the readout layer, which takes a dot product of the weights from the ridge regression with the node and bias. The value of the dot product produces a morphable logic gate bit stream.

Bits, which are encoded as sinusoids, are sent to the adaptive oscillator sequentially. It should be noted that the adaptive oscillator effectively "remembers" a previous value to operate on the current value for binary operations, such as the AND and OR tasks.

III. MODIFIED INFORMATION RATE

A modified version of Shannon's information rate is a relevant metric for logic gates, as the root-mean-square error is poorly defined for these logical tasks. The modified version is defined in terms of the original information rate [39], where the "sent" and "received" signals are replaced with the correct and predicted output of the gate, respectively.

The Shannon entropy $H(x)$ is the amount of information in the output. It should be noted that bits being sent to the logic gates are randomly chosen as "True" or "False" values with equal probability, but some of the gates do not have an equal probability of being "True" or "False." For instance, the NOT gate preserves the equal probability distribution, but the AND gate is "True" only 25% of the time (e.g., when both inputs are "True"). This unequal probability affects the upper limit on the information rate for different gates. For a bit i from the target of the gate and a bit j from the prediction of the gate, the Shannon entropy is defined as

$$H(x) = -\sum_i p_i \log_2(p_i), \tag{4}$$

where $p_i(j) = p(j \mid i) = \frac{p(i,j)}{\sum_j p(i,j)}$ is the conditional probability.

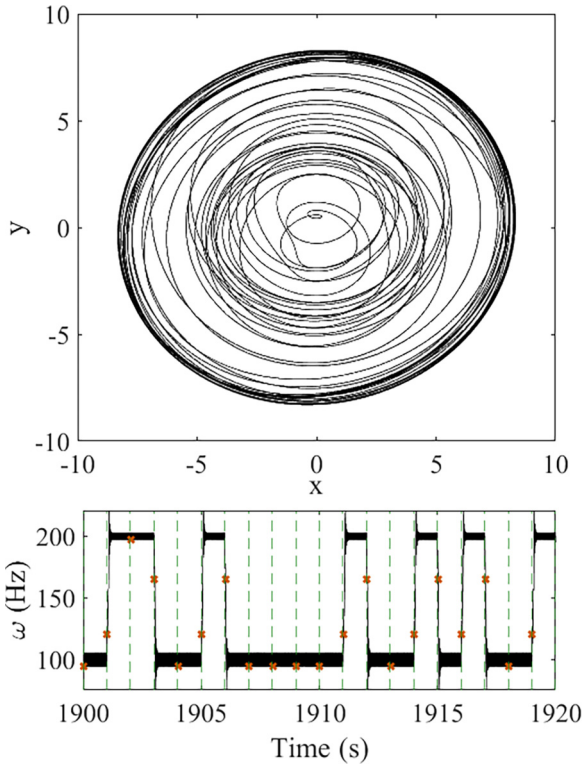

FIG. 2. The phase portrait of the x and y states are shown on the top. A portion of the response of the ω state (bottom) is plotted for reference. The green dashed lines correspond to the start of a clock cycle (e.g., these vertical lines delineate the pseudoperiods), and the orange $\times$'s are the nodal value components. Here, $\Omega_{\text{FALSE}} = 100$ Hz, $\Omega_{\text{TRUE}} = 200$ Hz, $T_p = 1$, $\mu = 1$, $k_x = 1000$, $k_\omega = 10\,000$, $\delta = 0.02$ s.

The conditional entropy $H_y(x)$ is the probability of an incorrect calculation of the gate. The conditional entropy is defined in terms of a joint probability $p(i, j)$ as

$$H_y(x) = -\sum_{i,j} p(i, j) \log_2[p_i(j)]. \tag{5}$$

Thus, the modified Shannon's information rate is calculated as

$$\text{IR} = H(x) - H_y(x). \tag{6}$$

If the adaptive oscillator PRC functions correctly as a logic gate, the IR value of the target will be equal to the $H(x)$ value of the prediction.

IV. LOGIC GATES

To highlight the efficacy of this physical reservoir computer, the NOT, AND, and OR gates are demonstrated, without modifying the adaptive oscillator itself. Thus, the reprogrammability of this system is solely based on the machine learning that is applied to the nodal outputs.

The AND, OR, and NOT gates are shown Fig. 3. Since the bits are sent to the adaptive oscillator sequentially, the system must effectively "remember" one bit to calculate the OR and AND tasks. The transient behavior of the adaptive oscillator's

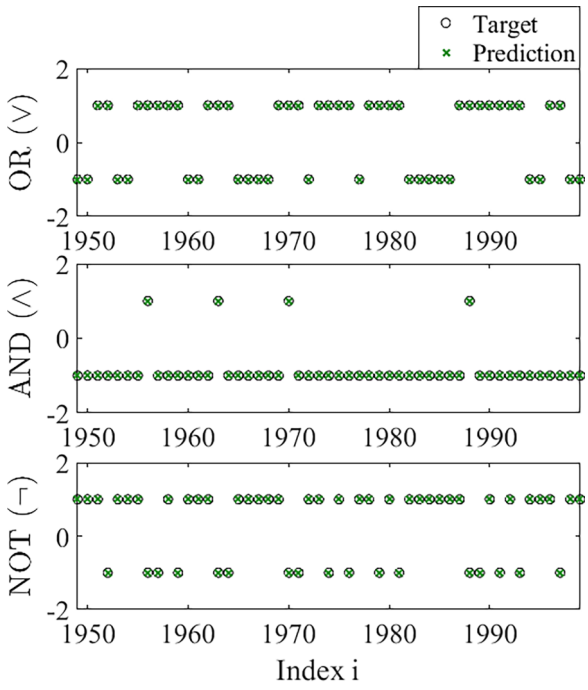

FIG. 3. Examples of the adaptive oscillator acting as an AND ($\wedge$), OR ($\vee$), and NOT ($\neg$) gate. For each task, the $H(x)$ and IR values were equal, which demonstrates that they work correctly [OR: $H(x) = 0.81$, IR $= 0.81$; AND: $H(x) = 0.74$, IR $= 0.74$; NOT: $H(x) = 1.0$, IR $= 1.0$]. Here, $\Omega_{\text{FALSE}} = 100$ Hz, $\Omega_{\text{TRUE}} = 200$ Hz, $T_p = 1$, $\mu = 1$, $k_x = 1000$, $k_\omega = 10\,000$, $\delta = 0.02$ s.

ω state provides a simple and effective method for collecting nodes from the system, without relying on time multiplexing.

In Figs. 4 and 5, the relationship between the computational ability of the adaptive oscillator (AO) PRC and the AO's system parameters are explored. As the Hopf oscillator is a nonlinear system, the computational ability is highly reliant on the oscillator's parameters, as can be seen in the bifurcated response in these figures. The normalized information rate $\frac{\text{IR}}{H(x)}$ is used to quantify the computational ability for the AND ($\wedge$) and OR ($\vee$), which both require the oscillator to remember the previous bit.

In Fig. 4, there is a rather quantized relationship between k_x and the computational ability, whereas the μ term has little effect on the computational ability. For the OR gate, there are only two quantized levels, whereas there are approximately three quantized levels of computational ability for the AND gate. A relatively large value of k_x (above approximately 560) results in both gates working correctly. In Fig. 5, both the k_x and k_ω terms affect the computational ability of the AO PRC. For the OR gate, there are still only two quantized levels. The AND gate has a more graded response before having a quantized jump to perfect computing. A combination of parameters, such as ($k_x = 1000$, $k_\omega = 10\,000$), results in perfect computation for both gates.

V. FIELD-PROGRAMMABLE ANALOG ARRAY CIRCUIT

A field-programmable analog array (FPAA) experiment is used to validate the performance of the adaptive

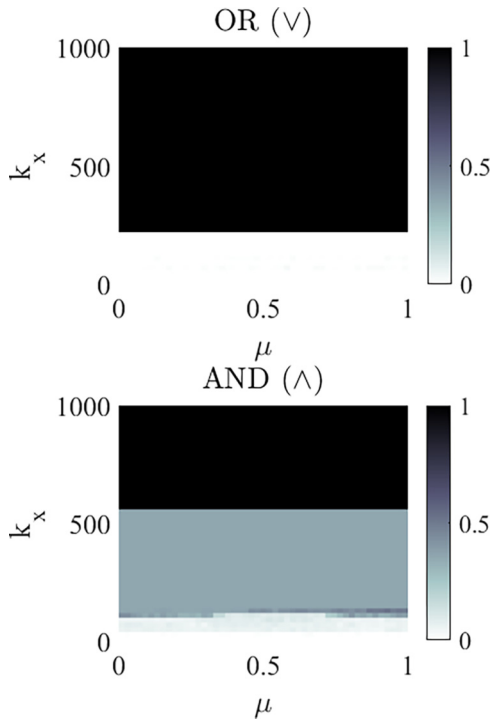

FIG. 4. The relationship between the μ term and the k_x term is explored for the AND ($\wedge$) and OR ($\vee$) gates. The normalized information rate $[\frac{\mathrm{IR}}{H(x)}]$ is shown as the color axis. Here, $\Omega_{\mathrm{FALSE}} = 100$ Hz, $\Omega_{\mathrm{TRUE}} = 200$ Hz, $T_p = 1$, $k_\omega = 10\,000$, $\delta = 0.02$ s.

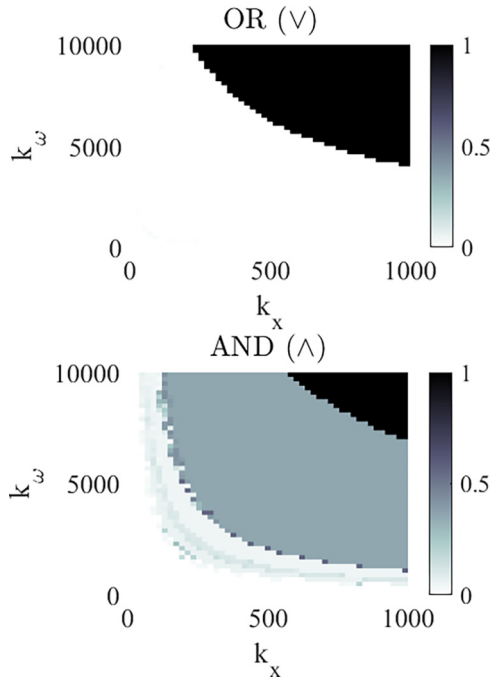

FIG. 5. The relationship between the k_x term and the k_ω term is explored for the AND ($\wedge$) and OR ($\vee$) gates. The normalized information rate $[\frac{\mathrm{IR}}{H(x)}]$ is shown as the color axis. Here, $\Omega_{\mathrm{FALSE}} = 100$ Hz, $\Omega_{\mathrm{TRUE}} = 200$ Hz, $T_p = 1$, $\mu = 1$, $\delta = 0.02$ s.

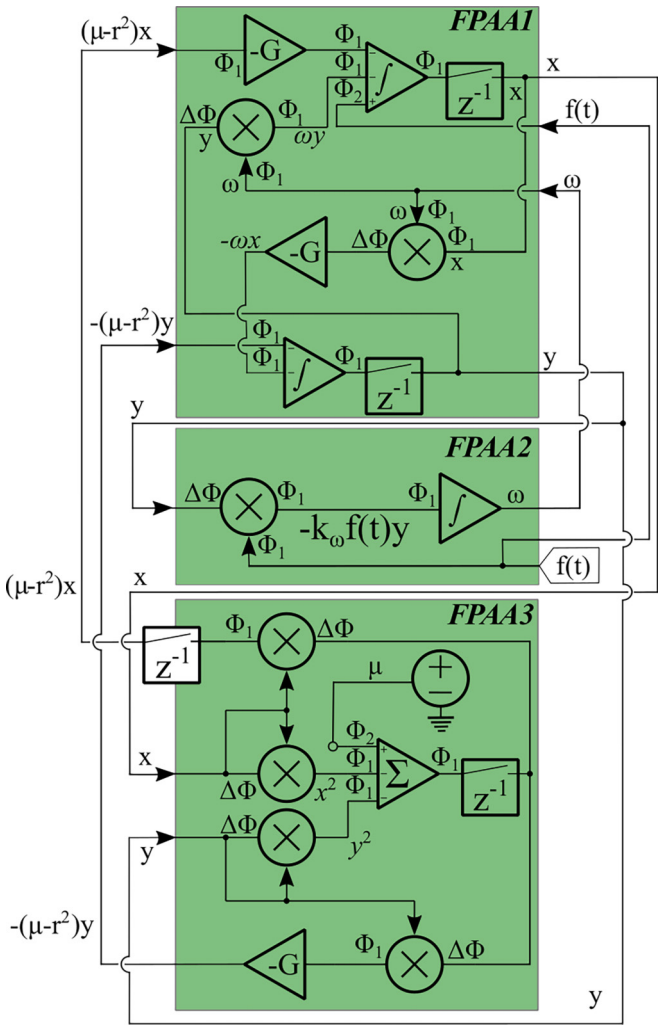

FIG. 6. The field-programmable analog array schematic is shown here. Three FPAA AN231E04 chips on an Anadigm QuadApex board (Anadigm, Paso Robles, CA) were used. The external forcing function $f(t)$ is sent to the FPAA from MATLAB via a National Instruments 9263 module.

oscillator PRC. FPAAs, which use switched-capacitor technology [40], are highly reconfigurable analog circuits. Many different nonlinear oscillators have been constructed with FPAAs, such as the van der Pol oscillator [12], Lorenz system [41], four-state adaptive oscillator [27], and a chaotic adaptive pendulum [33]. The adaptive oscillator PRC's circuit schematic is shown in Fig. 6.

The external signal $f(t)$ was generated in MATLAB. Next, it was sent to FPAA2 using a National Instruments 9263 module. A National Instruments 9201 module was used to collect the x, y, and ω states. Several configurable analog modules (CAMs) were used, including summation, multiplication, integration, dc voltage, and sample and hold. The r term denotes the limit cycle's radius, which is $r = \sqrt{x^2 + y^2}$.

In Fig. 7, the voltage from the ω state of the FPAA circuit is shown. The orange $\times$'s denote the nodes, which are sampled during the transient portion of the dynamics after each clock cycle (represented by vertical green dashed lines). As with the experiment, the adaptive oscillator PRC functions

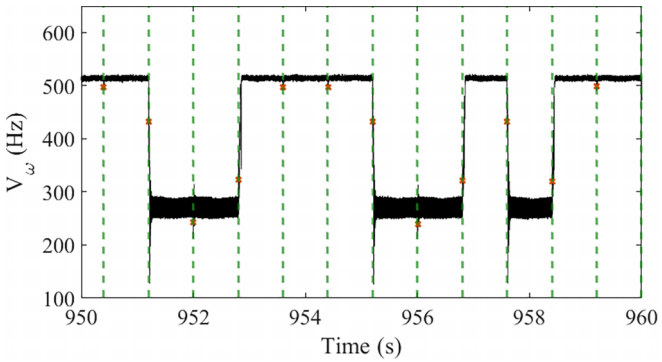

FIG. 7. A portion of the ω state from the FPAA experiment. The green dashed lines correspond to the clock, and the orange $\times$'s correspond to the nodes that are sampled during the transient portion of the response. Here, $\Omega_{\text{FALSE}} = 250$ Hz, $\Omega_{\text{TRUE}} = 500$ Hz, $T_p = 0.8$, $\mu = 2$, $k_x = 0.375$, $k_\omega = 0.125$, $\delta = 0.004$ s.

perform the OR, AND, and NOT tasks correctly, with the same calculated results as shown in Fig. 3. As with the simulation results, the nodes are $[1, \omega(nT_p + \delta)]$. The same regression procedure was used to train the PRC to reproduce the three considered logic gates.

VI. CONCLUDING REMARKS

In this paper, the adaptive oscillator is proposed as an analog reprogrammable logic gate. The adaptive oscillator has particularly interesting dynamics, as it has plastically deformable states that can calculate and store information. The reprogrammability of this logic gate is due only to the machine learning method used to train the nodal response to a target, and the adaptive oscillator itself is unmodified. This methodology highlights that time multiplexing can be completely avoided for oscillator-based physical reservoir computers by utilizing the adaptive states' dynamic response as nodes.

An adaptive oscillator has been used as a physical reservoir computer without multiplexing. Further, the adaptive oscillator's unique learning ability is repurposed for generalized tasks, such as logic gates. Previously, only frequency-related tasks, such as analog frequency analysis or tracking a time-varying signal's principal frequency component, have been shown.

From a practical perspective, this oscillator-based PRC without multiplexing can provide a faster method of computation, as the number of virtual nodes (usually tens or hundreds of virtual nodes) can be reduced to a single node. This reduces the sampling requirements of the hardware. Since the transient response can be used to calculate the logic gates, the clock frequency can be considerably reduced as well, but a relatively long clock frequency was chosen here for visualization purposes. The adaptive oscillator physical reservoir computer is capable of complex artificial intelligence (AI) inference tasks, such as wake word detection, image classification, and chaotic time series prediction. By repurposing the adaptive oscillator's learning ability for logic tasks, the adaptive oscillator can provide a robust method for calculating logic gates without additional hardware considerations. Moreover, as gates can be calculated in parallel, more complex outputs can be achieved synergistically with only the hardware needed for the AI inference tasks. However, this method could also be highly consolidated as a high-performance, morphable, parallelizable logic gate architecture.

ACKNOWLEDGMENTS

M.R.E.U.S. and X.L. contributed technical expertise on machine learning and adaptive oscillators. E.P. invented and reduced to practice the adaptive oscillator PRC, ran simulations, and conducted experiments.

APPENDIX A: ECHO STATE PROPERTY

The echo state property (ESP) is often considered to be a necessary condition for a system to work properly as a PRC. Here, the ESP index is calculated to determine if the AO PRC has the echo state property, which is shown in Fig. 8. A PRC possesses the echo state property if the response of the reservoir is independent of the initial conditions. Thus, a PRC has the echo state property if the effect of the initial conditions on the reservoir quickly fades. An ESP index of zero implies that the PRC possesses the echo state property [42].

APPENDIX B: MEMORY CAPACITY

The memory capacity of the AO PRC should also be mentioned. Usually, longer memory is desirable for PRCs, which is accomplished through time multiplexing. For example, in Ref. [10], a nonadaptive Hopf oscillator was used as the reservoir, and the memory capacity was evaluated for a fourth-order parity task. For the nonadaptive Hopf oscillator,

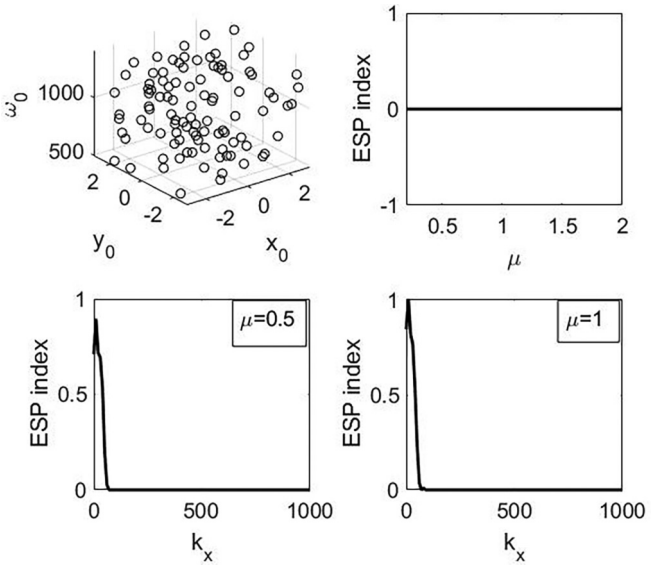

FIG. 8. The echo state property (ESP) index [42] is calculated for the AO PRC. Top left: The set of random initial conditions for the ESP index calculations. Top right: ESP index for the AO PRC for $k_x = 1000$ and $k_\omega = 10\,000$. Bottom left: ESP index for the AO PRC for $\mu = 0.5$ and $k_\omega = 10\,000$. Bottom right: ESP index for the AO PRC for $\mu = 1$ and $k_\omega = 10\,000$.

tens or hundreds of virtual nodes were collected using time multiplexing of the oscillator itself.

The desired goal is a parallelizable, morphable logic gate that can be implemented with the minimum number of sampled (virtual or physical) nodes, which can be reconfigured for AI inference tasks that use time multiplexing. The ω state is thus chosen as the only physical node. As can be visually observed in Fig. 2, the memory capacity of the AO is such that it can remember the single most recent input. It is noted that the AO PRC is structured to perform logic gate calculations, which necessitates one bit of memory stored in its dynamic states.

Here, only a single physical node, without time multiplexing, is sampled from the adaptive state. The AO PRC has less memory than the nonadaptive Hopf reservoir when used in this way. But, the trade-off is an approximately two orders of magnitude reduction in the number of required nodes. Thus, the described AO can be implemented at frequencies near the upper limit of the sampling rate for a dual AI inference processor that doubles as a general purpose processor.

[1] *Reservoir Computing*, edited by K. Nakajima and I. Fischer (Springer Nature, Singapore, 2021).

[2] H. Jaeger and H. Haas, Science **304**, 78 (2004).

[3] M. Lukoševičius and H. Jaeger, Comput. Sci. Rev. **3**, 127 (2009).

[4] B. Penkovsky, X. Porte, M. Jacquot, L. Larger, and D. Brunner, Phys. Rev. Lett. **123**, 054101 (2019).

[5] N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, Phys. Rev. E **91**, 020801(R) (2015).

[6] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Nat. Commun. **2**, 468 (2011).

[7] T. Natschläger, W. Maass, and H. Markram, Telematik **8**, 39 (2002).

[8] H. Jaeger, German National Research Center for Information Technology GMD Technical Report **148**, 13 (2001).

[9] M. R. E. U. Shougat, X. Li, T. Mollik, and E. Perkins, Sci. Rep. **11**, 19465 (2021).

[10] M. R. E. U. Shougat, X. Li, and E. Perkins, Phys. Rev. E **105**, 044212 (2022).

[11] M. R. E. U. Shougat, X. Li, S. Shao, K. McGarvey, and E. Perkins, Sci. Rep. **13**, 8719 (2023).

[12] M. R. E. U. Shougat and E. Perkins, Neuromorph. Comput. Eng. **3**, 024004 (2023).

[13] J. Moon, W. Ma, J. H. Shin, F. Cai, C. Du, S. H. Lee, and W. D. Lu, Nat. Electron. **2**, 480 (2019).

[14] M. R. E. U. Shougat, X. Li, T. Mollik, and E. Perkins, J. Comput. Nonlin. Dyn. **16**, 081004 (2021).

[15] S. Ghosh, A. Opala, M. Matuszewski, T. Paterek, and T. C. Liew, IEEE Trans. Neural Netw. Learn. Syst. **32**, 3148 (2020).

[16] A. Mizrahi, T. Hirtzlin, A. Fukushima, H. Kubota, S. Yuasa, J. Grollier, and D. Querlioz, Nat. Commun. **9**, 1533 (2018).

[17] J. Nokkala, R. Martínez-Peña, R. Zambrini, and M. C. Soriano, IEEE Trans. Neural Netw. Learn. Syst. **33**, 2664 (2021).

[18] J. Grollier, D. Querlioz, K. Y. Camsari, K. Everschor-Sitte, S. Fukami, and M. D. Stiles, Nat. Electron. **3**, 360 (2020).

[19] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, Phys. Rev. X **7**, 011015 (2017).

[20] B. Barazani, G. Dion, J.-F. Morissette, L. Beaudoin, and J. Sylvestre, J. Microelectromech. Syst. **29**, 338 (2020).

[21] M. R. E. U. Shougat, S. Kennedy, and E. Perkins, IEEE Sensors Lett. **7**, 6002304 (2023).

[22] S. Kan, K. Nakajima, Y. Takeshima, T. Asai, Y. Kuwahara, and M. Akai-Kasaya, Phys. Rev. Appl. **15**, 024030 (2021).

[23] L. Righetti, J. Buchli, and A. J. Ijspeert, *Proceedings of 3rd International Symposium on Adaptive Motion in Animals and Machines–AMAM 2005* (Verlag ISLE, Ilmenau, 2005).

[24] X. Li, M. R. E. U. Shougat, T. Mollik, A. N. Beal, R. N. Dean, and E. Perkins, J. Appl. Phys. **129**, 224901 (2021).

[25] L. Righetti, J. Buchli, and A. J. Ijspeert, Open Cybern. Syst. J. **3**, 64 (2009).

[26] X. Li, M. R. E. U. Shougat, S. Kennedy, C. Fendley, R. N. Dean, A. N. Beal, and E. Perkins, PLoS One **16**, e0249131 (2021).

[27] X. Li, M. R. E. U. Shougat, T. Mollik, R. N. Dean, A. N. Beal, and E. Perkins, Rev. Sci. Instrum. **94**, 035103 (2023).

[28] N. J. Corron, Chaos, Solitons Fractals **158**, 111991 (2022).

[29] M. F. Daqaq, R. Masana, A. Erturk, and D. D. Quinn, Appl. Mech. Rev. **66**, 040801 (2014).

[30] F. Cottone, H. Vocca, and L. Gammaitoni, Phys. Rev. Lett. **102**, 080601 (2009).

[31] S. C. Stanton, C. C. McGehee, and B. P. Mann, Physica D **239**, 640 (2010).

[32] X. Li, P. Kallepalli, T. Mollik, M. R. E. U. Shougat, S. Kennedy, S. Frabitore, and E. Perkins, Mech. Syst. Signal Process. **179**, 109361 (2022).

[33] L. Xiaofu, B. Aubrey, D. Robert, and E. Perkins, Chaos Theory Appl. **5**, 11 (2023).

[34] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, Neuromorph. Comput. Eng. **2**, 032002 (2022).

[35] K. Murali, S. Rajasekar, M. V. Aravind, V. Kohar, W. L. Ditto, and S. Sinha, Philos. Trans. R. Soc. A **379**, 20200238 (2021).

[36] S. Sinha and W. L. Ditto, Phys. Rev. Lett. **81**, 2156 (1998).

[37] A. Yao and T. Hikihara, Appl. Phys. Lett. **105**, 123104 (2014).

[38] A. Yao and T. Hikihara, Int. J. Non-Linear Mech. **94**, 406 (2017).

[39] C. E. Shannon, Bell Syst. Tech. J. **27**, 379 (1948).

[40] H. Kutuk and S.-M. Kang, *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96* (IEEE, New York, 1996), Vol. 4, pp. 41–44.

[41] E. Günay and K. Altun, Turkish J. Electr. Eng. Comput. Sci. **26**, 1812 (2018).

[42] C. Gallicchio, *27th European Symposium on Artificial Neural Networks* (Bruges, Belgium, 2019), pp. S2019-76.