EMESN: an Extended MOSFET Reservoir Computing Architecture for Echo State Networks with Hardware-Software Co-Optimization

Haoyuan Li[1,2], Masami Utsunomiya[2], Ryuto Seki[2], Takashi Sato[2], Feng Liang[1]

[1]School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China
[2]Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
jeword01@gmail.com, mutsunomiya, rseki@easter.kuee.kyoto-u.ac.jp,
takashi@i.kyoto-u.ac.jp, fengliang@xjtu.edu.cn

Abstract—This paper presents *EMESN*, an extended MOSFET hardware reservoir–computing architecture for time-series tasks. A pulse-based crossbar exploits intrinsic threshold-voltage variation to realize fixed random weights, while multi-mask mapping, ADC-range tuning and genetic optimization jointly enhance inference accuracy. Evaluations on eight public datasets demonstrate accuracy gains of up to 10.4 points and a 5× reduction in accuracy standard deviation, all with markedly lower static power than previous MOS-ESN designs.

I. Introduction

Reservoir Computing (RC) is a neuromorphic machine-learning framework in which low-dimensional temporal signals are projected into a high-dimensional state space by a fixed nonlinear network called the reservoir [1]. Because the reservoir weights are randomly initialized once and then frozen, training reduces to fitting a linear read-out—typically with ridge or logistic regression—thus eliminating back-propagation through time [2]. This train-once, read-out-only paradigm sharply lowers computation and energy consumption, making RC attractive for edge devices with tight resource budgets [3].

Among existing RC variants, the Echo State Network (ESN) is the most widely studied [4]. An ESN employs a sparsely connected reservoir stored as a sparse weight matrix; since that matrix never changes after initialization, ESNs map naturally onto hardware, especially analog implementations [3].

Driven by the demand for energy-efficient inference, researchers have begun to realize reservoir dynamics directly in hardware [5]. Memristive crossbars, photonic delay lines and spin-torque nano-oscillators have all been demonstrated [6, 7, 8]. MOSFET technology, however, combines mature fabrication, high yield and intrinsic threshold-voltage variation that supplies weight diversity without extra programming [9].

Although reservoir weights are fixed, software studies show that reconfiguring connectivity can still boost accuracy. Particle-swarm optimization (PSO) and the K2 hill-climbing algorithm have been used to prune or re-weight links, yielding measurable gains [10, 11]. In physical RC, by contrast, device conductances are hard-wired, so the problem becomes a discrete search over mask patterns. Continuous methods such as PSO lose effectiveness, and general graph-learning techniques are impractical because they rely on differentiable edge-loss functions that cannot be evaluated in situ [12].

EMESN, proposed in this work, is an extended MOSFET reservoir-computing system enhanced by bespoke optimizations. Our contributions are as follows:

1. **Pulse-driven MOSFET reservoir:** We adopt LMESN, an leakage-based low-power analog reservoir architecture proposed in another work, where leakage currents drive the state dynamics and threshold-voltage variation provides diverse fixed weights.

2. **Multi-mask mapping:** We pack a large logical reservoir onto a compact MOSFET array, increasing utilization and enabling larger effective reservoirs—and therefore higher accuracy—without enlarging silicon area.

3. **Hardware–software co-optimization:** By tuning the ADC lower quantization bound ($V_{\min}$), refining reservoir masks and applying a genetic algorithm suited to our discrete hardware search space, we improve accuracy and reduce variance across diverse time-series benchmarks.

The paper is organized as follows. Section II reviews the concept of ESN and existing works of hardware ESN based on memristors and MOSFETs. Section III introduces the hardware architecture of LMESN we based on, then discusses possible optimizations including multi-mask, ADC range and mask optimization. In section IV, we evaluate the proposed framework using variety of datasets, and section V concludes the paper.

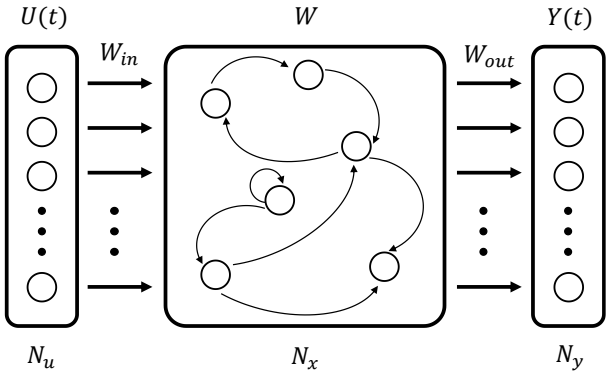

Fig. 1. General Structure of ESN.

II. Echo State Network

A. ESN Introduction

Fig. 1 sketches the overall architecture of ESN. The system consists of three layers: an input layer, a reservoir layer, and a read-out layer, whose weight matrices are denoted by W_{in}, W, and W_{out}, respectively. At time step t, the input vector $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ is first mapped through $W_{\text{in}} \in \mathbb{R}^{N_x \times N_u}$ and injected into the reservoir. The reservoir state $\mathbf{x}(t) \in \mathbb{R}^{N_x}$ is then updated according to

$$\mathbf{x}(t) = f_{\text{act}}\big(W_{\text{in}}\mathbf{u}(t) + W\mathbf{x}(t-1)\big), \tag{1}$$

where $W \in \mathbb{R}^{N_x \times N_x}$ is the recurrent reservoir matrix, and f_{act} is the activation function applied element-wise. The read-out layer produces the network output via

$$\mathbf{y}(t) = W_{\text{out}}\mathbf{x}(t), \tag{2}$$

with $W_{\text{out}} \in \mathbb{R}^{N_y \times N_x}$.

For a length-T input sequence the simplest strategy is to regard the output at the final time step as the prediction for the entire sample,

$$\mathbf{y}_{\text{seq}} = W_{\text{out}}\mathbf{x}(T). \tag{3}$$

However, when T is large, this 'last state' approach can discard information because the fading memory of the reservoir cannot capture very long dependencies. A common alternative is to average the reservoir states over the whole sequence,

$$\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}(t), \qquad \mathbf{y}_{\text{seq}} = W_{\text{out}}\bar{\mathbf{x}}, \tag{4}$$

which empirically preserves more temporal features for long inputs [16].

B. MOSFET ESN

In [9] a MOSFET-based hardware ESN (MOS-ESN) was proposed. The design exploits the inherent threshold-voltage variation of MOSFETs and represents signed weights with a differential pair operating in the linear region. Column currents are summed and sensed by a source-amplifier that converts the total current into a voltage, which is then broadcast to the next row and reconverted to a current.

This scheme, although elegant, suffers from two main drawbacks. First, operating the transistors in the linear region produces relatively large drain currents, so power dissipation scales unfavorably with the input dimension. Second, because the output voltage is fed directly into the reservoir at the next time step—without any ADC conversion—intermediate reservoir states cannot be stored or revisited. The classifier must therefore rely solely on the final state at $t = T$, which degrades inference accuracy for long sequences.

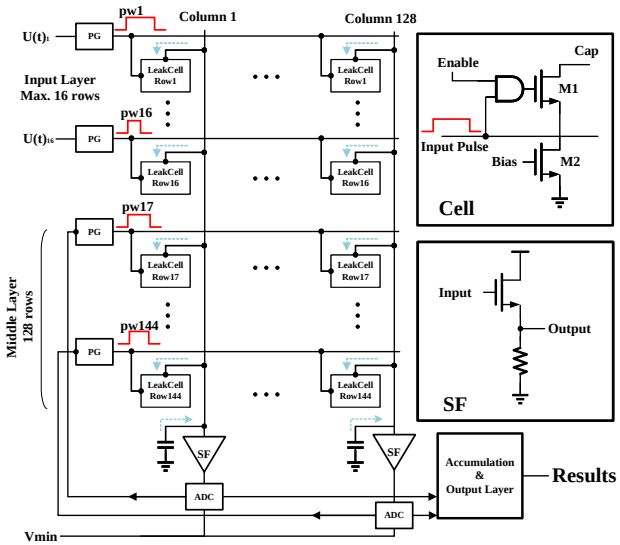

Fig. 2. Hardware architecture of LMESN.

III. EMESN

In this section we review LMESN, a crossbar array for ESN computation based on MOSFETs, with leakage current representing outputs, comprehensively explained in [13]. The proposed crossbar array acts as the hardware reservoir in between the input signals and software output layer. Given the importance of randomness in ESNs, we exploit the intrinsic variability in MOSFET threshold voltages. The resulting leakage drain currents are used to represent the input and reservoir weights, W_{in} and W, respectively.

A. Hardware Architecture

Hardware architecture of LMESN is illustrated in Fig. 2. The cell array consists of 144 × 128 cells (correspond to the LeakCell units shown in Fig. 2), where the first 16 rows are designated for storing the input weight

matrix $\mathbf{W}_{\text{in}}$, and the remaining 128 rows correspond to the reservoir matrix $\mathbf{W}_{\text{res}}$, defining a reservoir of size 128×128.

In each complete computation cycle, the capacitors connected to each column are first pre-charged to a fixed voltage level. Subsequently, the external input $\mathbf{u}(t)$ and the internal reservoir state $\mathbf{x}(t-1)$ are encoded as pulse widths and applied to the gate terminals of corresponding MOSFETs in the array.

During this process, the voltage on the capacitors is gradually discharged through the activated MOSFETs. The resulting voltage drop ΔV of each capacitor is approximately proportional to the total number of activated cells, their individual on-state currents, and the pulse widths applied. This relationship can be expressed as:

$$\Delta V \propto \sum_{i \in \text{ON}} I_i \cdot pw_{\text{in},i}$$

where I_i is the drain current of the i-th MOSFET, and $pw_{\text{in},i}$ denotes the pulse width applied to that cell.

The updated capacitor voltage, representing the current reservoir state $\mathbf{x}(t)$, is first transformed into another output voltage by source follower (SF) representing the activation function, then digitized by a 6-bit ADC. These digital values are subsequently fed into a row-wise pulse generator (PG), where they are converted into pulse signals to serve as the recurrent input for the next cycle, along with the next external input $\mathbf{u}(t+1)$. The computation can be expressed as:

$$X_{\text{pulses}}(t+1) = F_{\text{ADC-PG}}(f_{\text{SF}}(V_{\text{pre}} - \Delta V)) \qquad (5)$$

where $F_{\text{ADC-PG}}$ denotes the mathematical representation of ADC and PG. After all time steps T have been processed, the digital reservoir states collected at each time step are averaged to form a representative reservoir feature vector. This averaged feature is finally passed through the output layer to compute the overall output of the system.

Each cell consists of two NMOS transistors, as illustrated in Fig. 2. The gate of M_1 is driven by a two-input AND gate; its inputs are the row's input pulse and an *enable* stored locally, so M_1 conducts only when the cell is scheduled to participate in the current computation. Because the reservoir matrix W in an ESN is sparse, we employ a 128×128 Boolean *mask* that specifies which cells are enabled. This mask can be customized to optimize overall performance, as will be discussed in the following section. Transistor M_2 is biased by a low gate voltage V_{bias}. Consequently, even when a cell is enabled the drain current remains in the leakage regime, providing the desired ultra-low-power operation. This effectively limits the total leakage current across the array during each computation cycle, thereby minimizing energy dissipation without compromising functional performance. Moreover, due to inherent device-level variability—particularly the variation in threshold

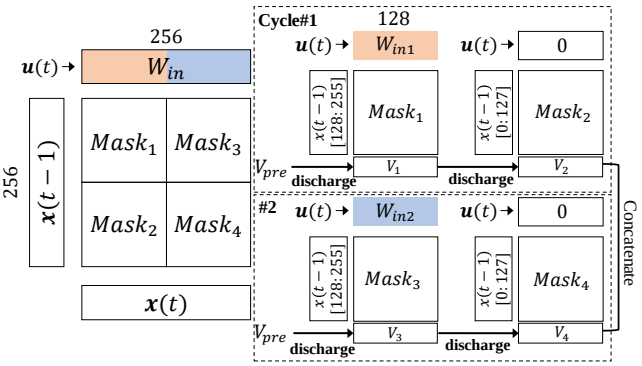

Fig. 3. Multi-mask method for large reservoir matrices.

voltage (V_{th}) across MOSFETs—the reservoir inherently exhibits stochastic behavior. This intrinsic randomness contributes positively to the diversity of the reservoir dynamics, which is beneficial for enhancing the expressive capacity of the system.

Compared to memristor-based implementations and our previously proposed MOS-ESN architecture, the LMESN system offers several advantages[3, 9]. MOSFET technology is more mature, cost-effective, and process-stable than memristors. Moreover, while memristors require additional programming to introduce device variability, MOSFETs inherently exhibit threshold voltage variation, providing randomness without costly initialization and maintaining reservoir diversity. Unlike our earlier MOS-ESN design, LMESN operates with significantly lower output current, which reduces dynamic power consumption and facilitates the integration of additional functionalities. Finally, the reservoir states are digitized and retained at each time step, improving temporal accuracy during recurrent processing and enabling future exploration of diverse reservoir representations based on stored digital features.

B. *Multi-mask for Larger ESN*

As previously described, each cell in the array includes an enable signal. Given the inherent sparsity of the reservoir matrix, we can represent the on/off state of each cell using a binary 128×128 switching matrix, referred to as a *mask*. This mask determines which devices are active during computation.

For more complex tasks, a larger reservoir often yields better expressive power. To construct reservoirs larger than the fixed 128×128 hardware array area-efficiently, we propose a multi-mask-based approach that decomposes a larger reservoir matrix into multiple smaller sub-blocks. As illustrated in Fig. 3, a reservoir matrix of size $K \cdot 128 \times K \cdot 128$ is divided into K^2 blocks, where K is an integer scaling factor (e.g., $K = 2$ for a 256×256 reservoir). We generate K^2 distinct masks and sequentially

map each of them onto the array for computation following a predefined schedule. Since the input matrix also needs to scale with K, the input vector is partitioned into K segments and applied over K consecutive time steps accordingly.

To maximize reservoir diversity, we design the masks such that their activated positions are as different as possible. For the input matrix, recall that the array contains 16 dedicated rows for input encoding. If the input dimension is less than $16/K$, we assign different input segments to different rows across the K time steps. If the input dimension exceeds $16/K$, we group overlapping rows into K input masks such that the union of all masks reconstructs a complete all-one input map.

For the reservoir matrix, which should be sparse, we ensure that the K^2 reservoir masks are mutually disjoint. This design choice increases structural variance across masks, which improves the dynamical richness of the overall reservoir system.

To reduce energy overhead from repeated ADC usage, we optimize the conversion schedule within each mask cycle. Instead of performing an ADC operation after every sub-step, we pre-charge the capacitors once at the beginning of a mask cycle and perform ADC readout only after all computations on that column are completed, as demonstrated in Fig 3. As discussed earlier, the leakage current is sufficiently small in LMESN cells to prevent significant voltage decay, thereby avoiding output saturation and preserving computational integrity throughout the cycle.

C. Dynamics Tuning and Optimization

The spectral radius is a crucial factor in the behavior of ESNs. If the spectral radius of the reservoir matrix is significantly smaller or greater than 1, the dynamic responses tend to become either vanishing or explosive, thus losing the diversity required for expressive reservoir computation.

In our hardware implementation, the reservoir matrix is implicitly defined by the device-level variability of MOSFETs, making it impossible to directly calculate its spectral radius. Moreover, combined with the complexities introduced by SF, ADC and PG, an explicit calculation of the spectral radius becomes intractable.

To address this, we focus on controlling the quantization range of the ADC, particularly its lower bound. In our q-bit ADC and pulse generator pipeline, the pulse output can be formulated as:

$$N_{\text{out}} = \left\lfloor \frac{V_{\text{state}} - v_{\min}}{v_{\max} - v_{\min}} \cdot (2^q - 1) \right\rfloor$$

where $v_{\min}$ and $v_{\max}$ define the ADC quantization range, N_{out} represents the number of output pulses, and q represents the ADC resolution. By tuning $v_{\min}$, we effectively adjust the dynamic sensitivity of the reservoir states, which in turn affects the final accuracy.

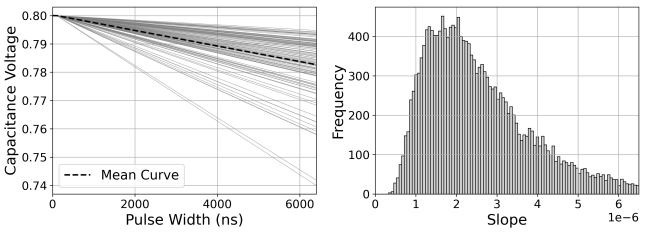

Fig. 4. Voltage-drop/pulse-widths relationship of single cells and slope distribution of all cells.

In software-based ESNs, the exact numerical values of the reservoir matrix are less critical, as long as the matrix meets sparsity, zero-mean, and desired spectral radius conditions. However, in hardware-based ESNs, once the chip is fabricated, each node's response is fixed, making the quality of the reservoir matrix task-dependent. For a given task or dataset, certain masks tend to result in higher inference accuracy on average, when evaluated across multiple trials or random seeds.

A prior study proposes using PSO to optimize such mask matrices [10]. In this approach, a continuous solution space is mapped to a binary mask via thresholding. However, PSO tends to converge prematurely to local optima, and its design—relying on disconnected subspaces created by thresholding—makes it difficult to scale in high-dimensional scenarios. The original work used 3000 optimization iterations, which is computationally expensive for large datasets and often not cost-effective.

To overcome these limitations, we propose using a Genetic Algorithm (GA) for mask optimization. Furthermore, leveraging the controllability introduced by $v_{\min}$, we augment the binary mask vector (of dimension N_x^2) with the quantization parameter $v_{\min}$ as an additional dimension, forming an optimization vector of dimension $N_x^2 + 1$.

Unlike PSO, where particles are updated based on velocity in continuous space, GA evolves the population through crossover and mutation, which is naturally well-suited for discrete search spaces. In the previous PSO work, sparsity is not controlled during optimization iterations. We propose to use special-designed mutation and crossover methods to control the sparsity of the reservoir matrices.

IV. Evaluation

In this chapter, we evaluate the performance of the EMESN system. For all experiments described in the following subsections, the evaluation follows a consistent procedure, as will be introduced in Seki Ryuto's paper. First, device-level characteristics are obtained through HSPICE-based simulations, which reflect the physical behavior of each MOSFET-based cell. These parameters are then used to perform a behavioral-level circuit simulation

TABLE I
PERFORMANCE COMPARISON UNDER DIFFERENT RESERVOIR SIZES AND MASK STRATEGIES

		Accuracy (%)			
Size	**128**	**256**		**384**	
Dataset		D	O	D	O
LIB	52.1	**56.7**	47.7	**58.3**	50.0
SS	92.0	**94.0**	92.1	**95.3**	92.9
AWR	86.4	**93.3**	89.0	**96.7**	92.3
MSH	85.4	**90.3**	86.8	**96.1**	88.9
PD	87.5	**89.7**	87.9	**91.1**	86.8
SAD	84.4	**92.0**	87.0	**95.9**	89.0

implemented in Python, emulating the reservoir dynamics over time. Based on the simulated reservoir states and training labels, the output layer weights are computed using Ridge regression [14]. The trained output layer is subsequently used to evaluate the model's performance on the test dataset. All simulations and evaluations are conducted on a workstation equipped with an Intel Core i9-13900 processor. All datasets and their full names are provided in [15].

A. Device Measurements

We begin by performing device-level simulations to characterize the voltage-drop/pulse relationship of the proposed cells, which demonstrates the intrinsic randomness required for effective ESN computation. Monte Carlo simulations were conducted using commercial 22 nm process design kit (PDK), and the output current of 100 randomly sampled cells is shown in Fig. 4. As observed, each cell exhibits a relatively stable linear relationship between input pulse-width and the resulting voltage drop, while maintaining noticeable variation across different cells.

B. Optimization Results

We evaluated and analyzed the effectiveness of the optimization strategies proposed in the previous section. First, we focus on the multi-mask strategy, where we examine training and inference results across several datasets using reservoir matrix sizes of 128×128, 256×256, and 384×384 on a fixed hardware array of 128×128. To further validate the benefit of sub-mask diversity, we test two different configurations: one with a 100% of overlap between masks, and the other with completely disjoint masks, noted as O and D, respectively. As shown in Table I, increasing the overall reservoir size with the proposed multi-masking strategy consistently improves the classification accuracy. However, when sub-masks share significant overlap, the performance gain is less pronounced, confirming that diversity among sub-masks plays a key role.

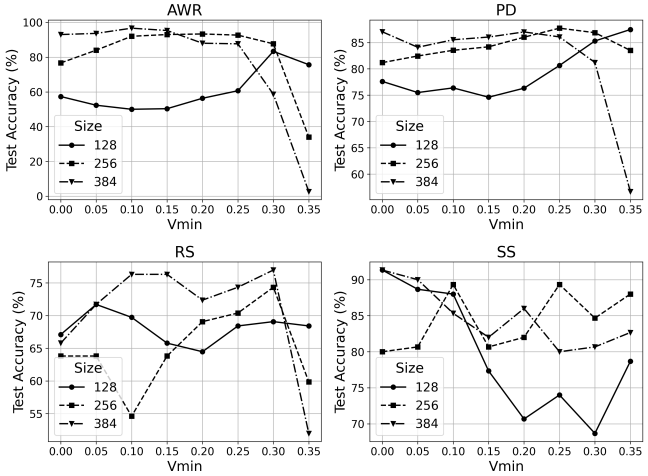

Fig. 5. Accuracy-V_{min} relationship of four different datasets.

TABLE II
ACCURACY COMPARISON BEFORE/AFTER GA/PSO OPTIMIZATION

		Accuracy (%)					
Dataset	**Soft. [16]**	**W/O Opt**		**PSO [10]**		**GA**	
		Mean	Std.	Mean	Std.	Mean	Std.
LIB	66.4	52.1	2.12	57.7	1.30	**61.7**	**1.28**
ECG	90.0	78.4	1.94	81.2	0.71	**87.2**	**0.76**
JV	98.4	95.6	0.45	97.1	0.34	**97.9**	**0.30**
RS	63.9	74.0	2.05	78.2	1.36	**84.8**	**1.31**
SS	86.7	92.0	1.62	95.8	0.23	**97.6**	**0.33**
AWR	89.6	86.4	1.41	89.3	0.57	**93.0**	**0.51**
MPOAG	63.6	71.4	0.82	72.9	0.58	**74.2**	**0.53**
MSH	90.6	85.4	1.50	89.1	0.66	**92.6**	**0.67**

Next, we assess the impact of tuning the ADC quantization lower bound $v_{\min}$ on the system's dynamic behavior and final accuracy. Experiments are conducted on the four different datasets, and the results are presented in Fig. 5. We observe that varying $v_{\min}$ significantly alters inference accuracy. Moreover, due to differences in input dimensionality, sequence length and reservoir size, the optimal $v_{\min}$ value is dataset-dependent. Therefore, $v_{\min}$ should be treated as a task-specific hyperparameter for maximizing performance.

Finally, as described earlier, we apply GA optimization to the reservoir mask structure. Table II summarises the results alongside the software-ESN baselines and PSO optimization results [10]. To highlight the necessity of such optimization, we begin by evaluating 100 randomly generated 128×128 masks. For each mask, we compute its best inference accuracy across a sweep of $v_{\min}$ values, and then record the corresponding mean and standard deviation.

We then apply GA to optimize the mask structure using a population size of 64 and 100 iterations—substantially

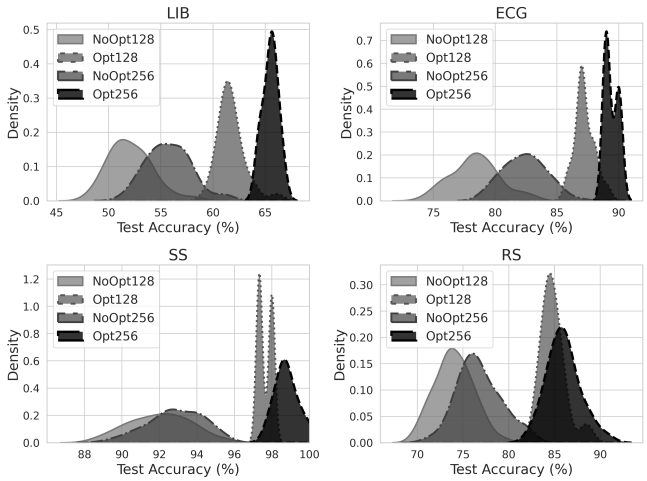

Fig. 6. Distributions of four datasets with/without optimization.

fewer than the number of cycles used in prior works such as [10]. Each dataset is independently tested with 30 optimization runs, and we report the mean and standard deviation of accuracy across these runs. Results show that the GA-optimized masks achieve both higher average accuracy and lower standard deviation compared to the randomly sampled masks, clearly demonstrating the effectiveness and necessity of the proposed optimization. Moreover, the optimized masks achieves accuracy fully comparable to the pure-software ESN and even surpasses it in cases where the software model overfits because of limited training data [16]. We also visualize the inference accuracy distributions for four representative datasets using kernel density estimation (KDE), as shown in Fig. 6. The results, covering both 128×128 and 256×256 reservoir sizes with our multi-mask methods, further highlight the superiority of the optimized mask structures.

V. Conclusion

We have introduced **EMESN**, an extended leakage-driven MOSFET ESN that exploits device threshold-voltage variation. Operating in the sub-nanoamp regime cuts static power, while pulse-based stimulation improves linearity and supports alternative reservoir encodings. A multi-mask strategy maps large virtual reservoirs onto a fixed-size array, boosting accuracy on demanding datasets. Finally, by analyzing the influence of ADC quantization range and jointly optimizing mask patterns and $V_{\min}$ with a genetic algorithm, we achieve up to 10 percentage point higher accuracy with $5\times$ reduction the accuracy standard deviation. These results show that careful mapping and quantization co-design can bring CMOS physical reservoirs to parity with, and sometimes beyond, their software ESN counterparts.

Acknowledgements

This work has been partially supported by JSPS KAKENHI grant Nos. 24K28052 and 23K18462.

References

[1] M. Yan, C. Huang, P. Bienstman, P. Tino, W. Lin, and J. Sun, "Emerging opportunities and challenges for the future of reservoir computing," *Nature Communications*, vol. 15, no. 1, p. 2056, 2024.

[2] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[3] X. Liang, J. Tang, Y. Zhong, B. Gao, H. Qian, and H. Wu, "Physical reservoir computing with emerging electronics," *Nature Electronics*, vol. 7, no. 3, pp. 193–206, 2024.

[4] C. Sun, M. Song, D. Cai, B. Zhang, S. Hong, and H. Li, "A systematic review of echo state networks from design to application," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 1, pp. 23–37, 2022.

[5] Y. Zhong *et al.*, "A memristor-based analogue reservoir computing system for real-time and power-efficient signal processing," *Nature Electronics*, vol. 5, no. 10, pp. 672–681, 2022.

[6] J. Moon *et al.*, "Temporal data classification and forecasting using a memristor-based reservoir computing system," *Nature Electronics*, vol. 2, no. 10, pp. 480–487, 2019.

[7] K. Vandoorne *et al.*, "Toward optical signal processing using photonic reservoir computing," *Optics Express*, vol. 16, no. 15, pp. 11182–11192, 2008.

[8] J. Torrejon *et al.*, "Neuromorphic computing with nanoscale spintronic oscillators," *Nature*, vol. 547, no. 7664, pp. 428–431, 2017.

[9] Y. Kume, S. Bian, and T. Sato, "A tuning-free hardware reservoir based on MOSFET crossbar array for practical echo state network implementation," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2020, pp. 458–463.

[10] Y. Xue, Q. Zhang, and A. Slowik, "Automatic topology optimization of echo state network based on particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 117, p. 105574, 2023.

[11] S. Bouazizi, E. Benmohamed, and H. Ltifi, "A novel approach of ESN reservoir structure learning for improved predictive performance," in *Proc. IEEE Symp. on Computers and Communications (ISCC)*, 2023, pp. 232–237.

[12] J. Liu, T. Sun, Y. Luo, S. Yang, Y. Cao, and J. Zhai, "Echo state network optimization using binary grey wolf algorithm," *Neurocomputing*, vol. 385, pp. 310–318, 2020.

[13] M. Utsunomiya, H. Murata, R. Seki, H. Li, H. Awano, and T. Sato, "LMESN: A Low-Power Hardware Reservoir Computing Architecture Based on MOSFET Leakage Variation," in *Proc. SASIMI*, 2025.

[14] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 93–100, 2009.

[15] M. Middlehurst, A. Ismail-Fawaz, A. Guillaume, C. Holder, D. Guijo-Rubio, G. Bulatova, L. Tsaprounis, L. Mentel, M. Walter, P. Schäfer, and A. Bagnall, "aeon: a Python toolkit for learning from time series," *Journal of Machine Learning Research*, vol. 25, no. 289, pp. 1–10, 2024.

[16] P. Tanisaro and G. Heidemann, "Time series classification using time warping invariant echo state networks," in *Proc. 15th IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, 2016, pp. 831–836.