Modular DFR: Digital Delayed Feedback Reservoir Model for Enhancing Design Flexibility

SOSEI IKEDA, Kyoto University, Japan

HIROMITSU AWANO, Kyoto University, Japan

TAKASHI SATO, Kyoto University, Japan

A delayed feedback reservoir (DFR) is a type of reservoir computing system well-suited for hardware implementations owing to its simple structure. Most existing DFR implementations use analog circuits that require both digital-to-analog and analog-to-digital converters for interfacing. However, digital DFRs emulate analog nonlinear components in the digital domain, resulting in a lack of design flexibility and higher power consumption. In this paper, we propose a novel modular DFR model that is suitable for fully digital implementations. The proposed model reduces the number of hyperparameters and allows flexibility in the selection of the nonlinear function, which improves the accuracy while reducing the power consumption. We further present two DFR realizations with different nonlinear functions, achieving 10x power reduction and 5.3x throughput improvement while maintaining equal or better accuracy.

CCS Concepts: • **Computer systems organization** → **Neural networks**.

Additional Key Words and Phrases: reservoir computing, delayed feedback reservoir (DFR), edge computing

1 INTRODUCTION

Reservoir computing (RC) is a machine learning method that uses a reservoir to nonlinearly transform inputs into high-dimensional vectors. In RC, the reservoir weights are not altered and the weights of the output layer that follows the reservoir are the target of learning [13], which allows efficient training. RC is considered suitable for time series processing because of its recurrent structure, which reflects past inputs. Because the weights of a reservoir are fixed, it can be implemented in hardware utilizing various physical phenomena.

A delayed feedback reservoir (DFR) [2] is a specific type of RC system. It is particularly suitable for hardware implementations because it can be compactly constructed with a single nonlinear element and a feedback loop [18]. Until now, hardware implementations of DFRs have been of two types: analog and digital [2]. In an analog implementation, only the nonlinear element of the reservoir or the nonlinear element and the feedback loop are implemented in an analog manner. However, the inputs and outputs are generally processed digitally, requiring a digital-to-analog converter (DAC) and an analog-to-digital converter (ADC). In addition, the time required for signal propagation through the feedback loop reduces the throughput. In comparison, digital

Authors' addresses: Sosei Ikeda, Kyoto University, Kyoto, Japan, sikeda@easter.kuee.kyoto-u.ac.jp; Hiromitsu Awano, Kyoto University, Kyoto, Japan, awano@i.kyoto-u.ac.jp; Takashi Sato, Kyoto University, Kyoto, Japan, takashi@i.kyoto-u.ac.jp.

implementations mimic the nonlinear behaviors of analog implementations in a fully digital manner. However, typical implementations involve division and power calculations that require significant hardware resources.

This paper proposes a novel interpretation of the DFR equation and a corresponding DFR model that enables an efficient fully digital implementation. The proposed interpretation relaxes the selection of the nonlinear function without increasing the number of hyperparameters, thereby simplifying parameter search and improving its efficiency. The proposed model is different from existing analog implementations, which require a DAC and an ADC, and which involve a delay associated with the feedback loop.

Based on this model, we propose identity and tanh DFRs, which use identity and hyperbolic tangent functions, respectively, as the pluggable nonlinear function. The identity DFR requires neither complex modules such as division nor exponentiation calculations, different from existing digital implementations. The tanh DFR provides flexibility to adjust the nonlinearity of the reservoir. In this study, software evaluation validated the time series prediction accuracy compared with those of nine existing machine learning methods. The tanh DFR was observed to outperform these state-of-the-art methods, and the identity DFR demonstrated comparable accuracy to them on most datasets. In addition, the identity DFR was evaluated in a hardware implementation. The hardware evaluation using a field-programmable gate array (FPGA) showed 10x and 5.3x improvements simultaneously in the throughput and the power consumption, respectively, compared to those of a hybrid FPGA–aplication-specific integrated circuit (ASIC) DFR [16].

The important contributions of this study can be summarized as follows:

(1) We developed a new interpretation of the DFR equation with a corresponding model called modular DFR that facilitates efficient design of a fully digital DFR.
(2) As practical examples, we developed identity and tanh DFRs, which use identity and hyperbolic tangent functions, respectively, as the pluggable nonlinear function.
(3) The effectiveness of the proposed DFRs was verified by comparing them with existing machine learning methods and DFR implementations using an FPGA.

The remainder of this paper is organized as follows. First, Section 2 reviews the concept and operation of RC and DFR. Section 3 proposes a new DFR model that enhances flexible design of a fully digital DFR and presents two practical examples of the model. Section 4 presents the evaluation of the proposed model from both software and hardware perspectives, and, finally, Section 5 concludes the paper.

2 RC

2.1 Concept of RC

RC consists of an input layer, a reservoir, and an output layer [13]. The input layer converts an input signal into a form that the reservoir can process. The reservoir subsequently transforms the signal from the input layer into a high-dimensional nonlinear vector called the *reservoir state.* Finally, the output layer performs pattern recognition based on a simple learning algorithm. Each layer is characterized by weights connecting the nodes in that layer. The weights of the input and reservoir layers are fixed, and the learning process only determines the weight parameters of the output layer.

RC is considered particularly suitable for processing time-series data because of its recurrent structure, which can reflect past inputs. In addition, the reservoir layer does not require learning and therefore can be implemented in fixed hardware utilizing various physical phenomena [18].
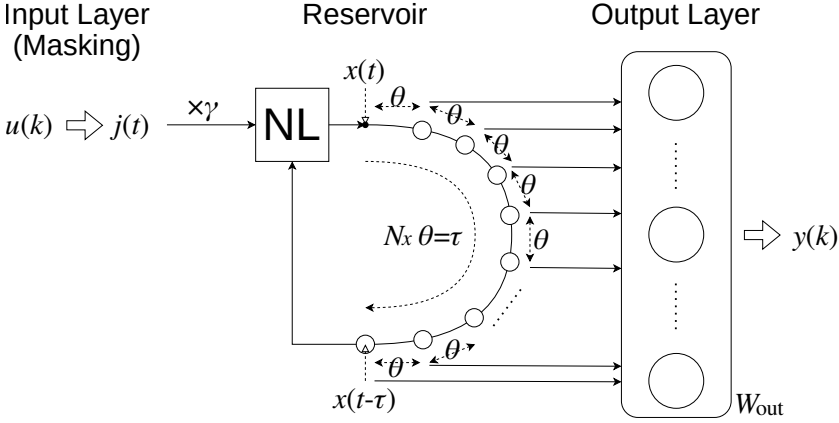
Fig. 1. Conceptual diagram of DFR [2]. The reservoir consists of a nonlinear element (NL) and a feedback loop with a total delay τ. The feedback loop comprises N_x virtual nodes with a time interval θ.

2.2 DFR

Figure 1 illustrates a conceptual diagram of a DFR. First, the input signal, $u(k)$, is time-multiplexed to $j(t)$ by masking and is input to the nonlinear element (NL) with the feedback signal. The nonlinear element is represented by the following delay differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = F(x(t-\tau), \gamma j(t)). \tag{1}$$

Consider that the virtual nodes are connected at a time interval θ on the feedback loop with a total delay τ. The signal values at the virtual nodes are the features, and they are collected as a vector of N_x elements as the reservoir state. Here, the following relationship holds:

$$N_x\theta = \tau. \tag{2}$$

The reservoir state is defined as follows:

$$\boldsymbol{x}(k) \equiv [x(k\tau - \theta), x(k\tau - 2\theta), \dots, x(k\tau - \tau)]. \tag{3}$$

The output of the DFR is obtained by linearly transforming the reservoir state at the output layer. The detailed operation is explained subsequently.

2.3 Analog implementation of DFR

The reservoir in the DFR has a relatively simple structure consisting of a nonlinear element and a feedback loop. Conversely, well-known RC methods, such as the echo state network, employ a recurrent neural network as the reservoir. A DFR is easier to implement in hardware than other RC methods [18]. Most existing hardware implementations of DFRs use analog circuits in the reservoir [2, 17].

A block diagram of an analog implementation of a DFR is shown in Fig. 2. First, the digital time series input, $u(k)$, sampled with a period τ, is converted to an analog signal $i(t)$. $i(t)$ is subsequently multiplied by the mask signal with a faster sample rate θ in the masking process (Fig. 3). Here, θ is the delay between the nodes in the reservoir. The length of the mask pattern is equal to the number of virtual nodes in the DFR, as expressed in Eq. (2). Each mask value is applied at each time interval θ. Fig. 3 shows the most common mask signals, +1 and −1; however, different masks can be used [2, 3].
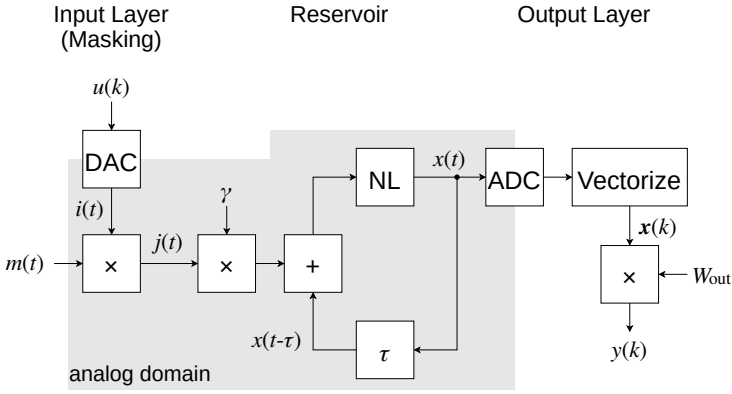
Fig. 2. Block diagram of DFR for analog implementation of a nonlinear element and a feedback loop [2]. The DAC and ADC are placed before masking and after the reservoir, respectively.
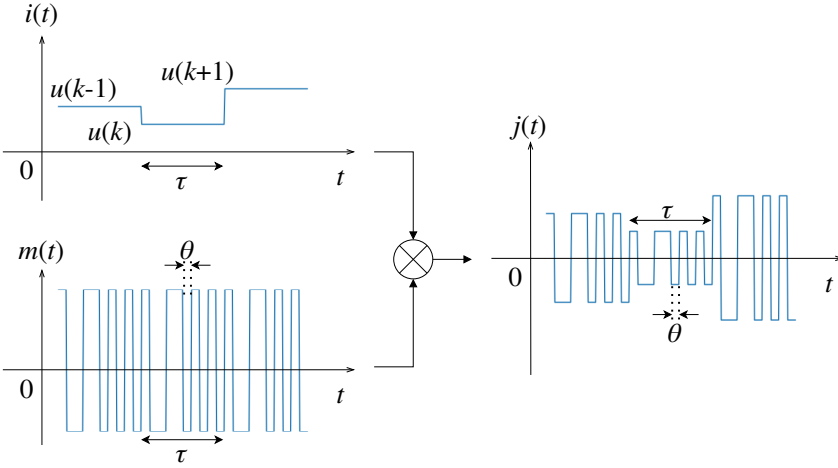

Fig. 3. Masking process. i is obtained by digital-to-analog conversion of input signal u. Signal i is constant at all τ. Mask signal m takes different values at all time intervals θ, and its period is τ. Input signal to reservoir is expressed as $j(t) = i(t) \cdot m(t)$.

$j(t)$ is subsequently scaled by γ, added to the reservoir feedback, and enters the nonlinear element (NL), which produces the output, $x(t)$. Mackey–Glass (MG) model [14] is commonly used as the NL [1, 2, 17]. Its operation is expressed as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t} x(t) = -x(t) + \eta f(x(t-\tau), \gamma j(t)), \tag{4}$$

$$f(x(t-\tau), \gamma j(t)) = \frac{[x(t-\tau) + \gamma j(t)]}{1 + [x(t-\tau) + \gamma j(t)]^p}, \tag{5}$$

where p is an adjustable parameter. $x(t)$ is sampled and converted to digital at each time interval θ. For each input data, sampling is performed N_x times, and the reservoir state, or the feature vector, is obtained, as expressed in Eq. (3). Simultaneously, $x(t)$ undergoes the feedback loop with a delay time τ and is added to $\gamma j(t)$. Note that Eq. (2) holds.

The output is obtained by linearly transforming the reservoir state at the output layer. In [11], the reservoir state, $\tilde{x}(k)$, is defined as follows:

$$\tilde{x}(k) \equiv [x(k), 1]. \tag{6}$$

The output is subsequently obtained by applying a linear transformation as follows:

$$y(k) = W_{\text{out}}\tilde{x}(k). \tag{7}$$

Various circuit designs with different ratios of analog to digital circuits exist. Examples include a primarily digital approach in which only the nonlinear element is implemented in analog, whereas the feedback loop is in the digital domain [2, 16]. In this approach, the DAC and the ADC in a DFR are placed before and after the nonlinear element, respectively. One common aspect of these implementations is that both DAC and ADC are essential. Although digital circuits may consume less power, power reduction of the converter circuits is significantly challenging. Additionally, obtaining an element of the reservoir state requires time θ, and obtaining the reservoir state for each input takes time τ. The conversion times of the ADC and the DAC further reduce the throughput.

2.4 Digital implementation of DFR

Fully digital implementations of a DFR also exist [1, 2]. In digital DFRs, the signal, $j(k)$, after the masking process is expressed as [2]

$$j(k) = mu(k), \tag{8}$$

where m is a mask vector of length N_x and whose elements are determined randomly and digitally. After masking, the nonlinear element and feedback loop of the DFR determine the reservoir state, $x(k)$.

Most implementations use the MG model as the nonlinear element. Assuming that f is constant for a short time θ, which is the time interval between the virtual nodes, the differential equation in Eq. (5) is solved for $x(t)$ as follows:

$$x(t) = x_0 e^{-t} + \eta(1 - e^{-t})f(x(t - \tau) + \gamma j(t)), \tag{9}$$

where x_0 is the initial value of $x(t)$ at each θ [2]. The value of the next virtual node can be expressed as $x(\theta)$. Assuming that the components of $x(k)$ and $j(k)$ are

$$x(k) \equiv [x(k)_1, x(k)_2, \ldots, x(k)_{N_x}], \tag{10}$$

$$j(k) \equiv [j(k)_1, j(k)_2, \ldots, j(k)_{N_x}], \tag{11}$$

$x(k)$ is obtained recurrently as follows:

$$x(k)_1 = x(k-1)_{N_x}e^{-\theta} + \eta(1 - e^{-\theta})f(x(k-1)_1 + \gamma j(k)_1), \tag{12}$$

$$x(k)_n = x(k)_{n-1}e^{-\theta} + \eta(1 - e^{-\theta})f(x(k-1)_n + \gamma j(k)_n). \ (n \geq 2) \tag{13}$$

Here, the initial value of the reservoir state is set to 0. The construction and operation of the output layer are the same as in an analog implementation.

Although the existing digital implementations avoid the use of a DAC and an ADC, the choice of the nonlinear function, f, is limited to those mimicking analog implementations, such as the MG model, which requires significant hardware resources.

3 NEW DIGITAL DFR MODEL ENHANCING DESIGN FLEXIBILITY

In this section, we propose a new DFR model suitable for digital implementations. The primary objective in its development was to improve the flexibility of choosing the nonlinear block in a digital DFR. The freedom to choose the nonlinear block simplifies the design and enhances the accuracy compared with existing digital DFRs. Although adding this extra design feature may
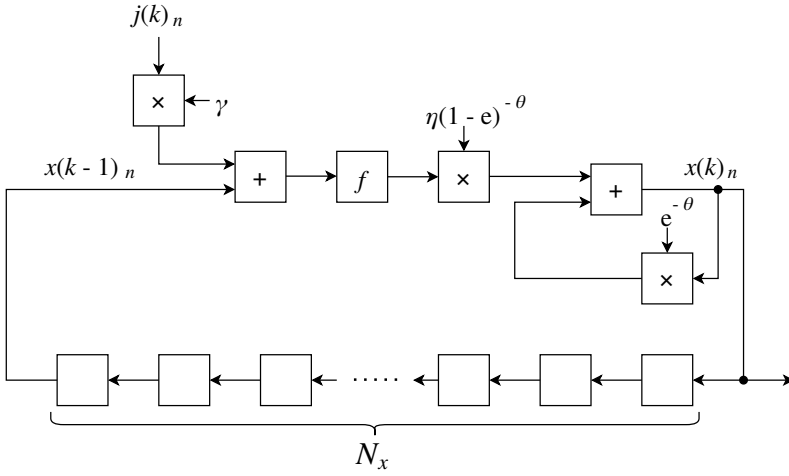
Fig. 4. Block diagram of reservoir processing based on the reformulation of the traditional digital DFR model. Block labeled "f" operates according to one-input, one-output function f in Eq. (5). Empty blocks are virtual nodes of the reservoir, which functions as shift registers.

seem to increase the design complexity, in this section, we subsequently show analytically that the hyperparameter, γ, for input scaling can be eliminated, which eventually simplifies the overall design. Finally, we present two examples of the new DFR model.

3.1 Modular DFR

Based on Eq. (13), $x(k)_n$ depends on $x(k)_{n-1}$, $x(k-1)_n$, and $j(k)_n$. The recursive relationship between the inputs, $x(k)_{n-1}$ and $x(k-1)_n$, with the output $x(k)_n$ poses a challenge for designing a digital nonlinear block of a DFR reservoir. To address this issue, we propose reformulation of the traditional model to use a one-input, one-output function f. The functionality of a reservoir in the digital domain can be represented, as shown in Fig. 4. This interpretation simplifies the design process by decomposing the operations of the nonlinear element of the reservoir into multiple blocks. This makes the design of the one-input, one-output function, f, independent of other parameters.

The original DFR reservoir requires adjustment of three parameters: γ, η, and θ. To mitigate the design complexity due to the flexibility in choosing the nonlinear function, we conducted an analytical study of the relationships between these parameters. We supposed that γ is scaled to $\alpha\gamma$ and x' is obtained accordingly. Furthermore, we assumed that the scaling relationship holds for the reservoir state at the previous time, $(k-1)$, expressed as follows:

$$\boldsymbol{x}'(k-1) = \alpha\boldsymbol{x}(k-1). \tag{14}$$

We consider the changes in the variable, x', and scaling of f to define function f' as follows:

$$f'(x) = \alpha f(\frac{x}{\alpha}). \tag{15}$$

Therefore, the first virtual node at time k satisfies scaling by α as follows:

$$
\begin{aligned}
x'(k)_1 &= x'(k-1)_{N_x}e^{-\theta} + \eta(1-e^{-\theta})f'(x'(k-1)_1 + \alpha\gamma j(k)_1)\\
&= \alpha x(k-1)_{N_x}e^{-\theta} + \alpha\eta(1-e^{-\theta})f(x(k-1)_1 + \gamma j(k)_1)\\
&= \alpha x(k)_1.
\end{aligned} \tag{16}
$$

A similar calculation yielded that the scaling relationship, $x'(k)_n = \alpha x(k)_n$ $(n \geq 2)$, holds for all other nodes. Because the relation in Eq. (14) also holds for $k = 1$ with zero as the initial value of $\boldsymbol{x}'$, by induction, this relation holds for all k as follows:

$$
\boldsymbol{x}'(k) = \alpha\boldsymbol{x}(k). \tag{17}
$$

Subsequently, the reservoir state produced by any input scaling with γ can be achieved equivalently by substituting f with f' in Eq. (15).

The above discussion also applies to the output layer. Ridge regression, a widely used method for training the output layer of a DFR, is performed as follows. First, let $d(k)$ be the target output for an input $u(k)$. We subsequently define the following:

$$
\tilde{X} = [\tilde{x}(1), \tilde{x}(2), \ldots, \tilde{x}(T)], \tag{18}
$$
$$
D = [d(1), d(2), \ldots, d(T)], \tag{19}
$$

where T is the number of training data. In ridge regression, W_{out} is derived as follows:

$$
W_{\text{out}} = D\tilde{X}^{\text{T}}(\tilde{X}\tilde{X}^{\text{T}} + \beta I)^{-1}, \tag{20}
$$

where I is an identity matrix of size $N_x \times N_x$ and β is a regularization parameter. Once more, considering the scaling of the input, $\tilde{X}$, and parameter β:

$$
\tilde{X}' = \alpha\tilde{X}, \tag{21}
$$
$$
\beta' = \alpha^2\beta, \tag{22}
$$

the corresponding W'_{out} to the input scaling is obtained as

$$
\begin{aligned}
W'_{\text{out}} &= D\tilde{X}'^{\text{T}}(\tilde{X}'\tilde{X}'^{\text{T}} + \beta'I)^{-1}\\
&= D\alpha\tilde{X}^{\text{T}}(\alpha^2\tilde{X}\tilde{X}^{\text{T}} + \alpha^2\beta I)^{-1}\\
&= \frac{1}{\alpha}W_{\text{out}}.
\end{aligned} \tag{23}
$$

Because the output, $y(k)$, is derived using Eq. (7), the same result as the input scaling can be achieved using the matrix in Eq. (23) with β multiplied by α^2. In summary, using the new model, called modular DFR in Fig. 5, the same solution space with the original DFR is obtained. The proposed model reduces the parameter space to be explored to two dimensions while enabling the selection of function f, making the optimization of a digital DFR easier and more efficient. A and B in Fig. 5 correspond to $\eta(1-e^{-\theta})$ and $e^{-\theta}$ in Fig. 4.

The most crucial aspect of a DFR design is the selection of the nonlinear element and associated parameters. In an analog implementation, selecting an element that satisfies the delay differential equation is sufficient. The proposed model simplifies the digital DFR design process, making it as straightforward as its analog counterpart. Instead of reproducing the recursive differential equation, we can more freely design the one-input, one-output function, f, with fewer parameters to learn. In addition, the processing in the proposed modular DFR, except that for f, is simple, comprising only constant multiplications, additions, and shifts. This makes it easier to implement low-power and high-throughput digital DFR using existing design tools.
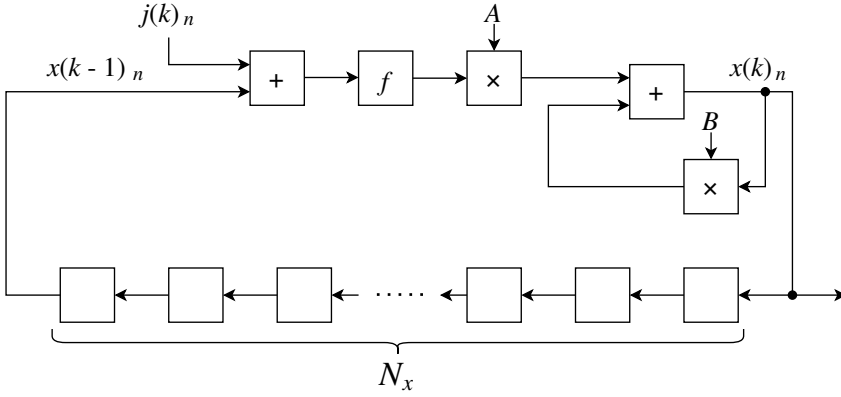
Fig. 5. Block diagram of the proposed modular DFR. Only two parameters, A and B, have to be optimized.

Fig. 6. Behavior of f in Mackey–Glass model. Parameter p for adjusting nonlinearity are 1 and 7, as used in [2].

3.2 Examples of proposed model

As examples of the proposed modular DFR, we present two DFRs that use identity or hyperbolic tangent functions as f.

Fig. 6 shows the shapes of the MG models as f, with $p = 1$ and $p = 7$ as used in [2]. The function, f, exhibits nonlinearity, the degree of which can be adjusted with p. [2] evaluated the performance of a DFR with the MG model using NARMA10 and spoken digit, which are time-series prediction and spoken word classification tasks, respectively. The ranges of $x + \gamma j$ to solve the tasks are depicted in Fig. 7 using the same parameters as in [2]. Notably, although f in the MG model exhibits nonlinearity, only a narrow region, which is considered to be mostly linear, is used in most tasks, including those mentioned above.

Based on this rationale, we propose two examples of the nonlinear function, f, of the developed modular DFR. Because both these examples are characterized by a linear region near the origin, we propose an identity function as f as the first and simplest example. In this case, since $f(x) = x$

Fig. 7. Output of function f with input range used to solve tasks in [2]. $p = 1$ and $p = 7$ correspond to NARMA10 and spoken digit tasks, respectively. Only weak nonlinearity was used in both examples.

Fig. 8. Input–output characteristics of the hyperbolic tangent function as f (Eq. (24)) for different α.

holds, the scaling of f expressed in Eq. (15) is unnecessary. The second example is a hyperbolic tangent function as f to better control the nonlinearity, which is expressed as:

$$f(x) = \alpha \tanh \frac{x}{\alpha} = \alpha \frac{e^{\frac{x}{\alpha}} - e^{-\frac{x}{\alpha}}}{e^{\frac{x}{\alpha}} + e^{-\frac{x}{\alpha}}}. \tag{24}$$

This formula considers the parameter scaling as expressed in Eq. (15). The shape of f when it is a hyperbolic tangent function for different α is plotted in Fig. 8. Increasing and dicreasing the value of α widen and narrow the linear section, respectively. Thus, by changing α, the nonlinearity of f can be adjusted.

4 EVALUATION

In this section, we present the performance evaluation of the proposed DFR models. First, we compare their accuracy with those of existing machine learning techniques. Subsequently, existing hardware implementations of DFRs are compared in terms of the accuracy, circuit size, power consumption, and throughput.

4.1 Comparison with existing machine learning techniques

To compare the proposed methods with existing machine learning techniques, we employed three time-series prediction tasks. Two of these involved typical chaotic systems: the Lorenz and Rössler systems. The third task was a real magnetic storm loop current index dataset, specifically the disturbance storm time (DST) dataset [4]. These tasks were the same as those used in the state-of-the-art chaotic time series prediction method, MOICBLS [22].

The Lorenz system is defined as:

$$
\begin{aligned}
\frac{dx}{dt} &= -a(x - y), \\
\frac{dy}{dt} &= -xz + cx - y, \\
\frac{dz}{dt} &= xy - bz,
\end{aligned}
\tag{25}
$$

where $a = 10$, $b = 8/3$, $c = 28$, $x(0) = 1$, $y(0) = 0$, and $z(0) = 1$ in the following experiments.

The Rössler system is defined as:

$$
\begin{aligned}
\frac{dx}{dt} &= -y - z, \\
\frac{dy}{dt} &= x + ay, \\
\frac{dz}{dt} &= b + z(x - c),
\end{aligned}
\tag{26}
$$

where $a = 0.15$, $b = 0.2$, $c = 10.0$, $x(0) = 1$, $y(0) = 0$, and $z(0) = 1$.

For both systems, the fourth-order Runge–Kutta method was used to generate 12,000 samples with a sampling interval of 0.01 s. Of these, the first 9,000 samples were used for training, and the remaining 3,000 were used for testing.

The DST index dataset, which has a sampling period of 1 h [8], was used in this study. Specifically, 4,000 samples from the tail of both the 2013 and 2014 data were used. Among these, the first 3,000 samples were allocated for training, whereas the remaining 1,000 were used for testing. Note that all the settings used in this study were consistent with those in [22].

The machine learning methods compared were as follows:

(1) random forest (RF) [20]
(2) extreme learning machine (ELM) [21]
(3) support vetor machine (SVM) [7]
(4) elman artificial neural network (Elman) [5]
(5) denoising autoencoder (DAE) [12]
(6) gradient boosting (GB) [9]
(7) echo state network (ESN) [10]
(8) broad learning system (BLS) [6]
(9) intergroup cascade BLS with multiobjective optimized parameters (MOICBLS) [22]

The prediction accuracies of these methods, as reported in [22] were used.

Table 1. Parameters of the DFRs used for comparison with existing machine learning techniques.

dataset	MG DFR			identity DFR		tanh DFR		
	γ	η	β	A	β	A	α	β
Lorenz x	0.02	0.001	1E-17	0.09	1E-5	0.1	10	1E-5
Lorenz y	0.04	0.5	1E-11	0.05	1E-7	0.15	4	1E-4
Lorenz z	0.02	0.04	1E-14	0.1	1E-6	0.07	15	1E-6
Rössler x	0.003	0.01	1E-16	0.04	1E-4	0.04	4000	1E-4
Rössler y	0.001	0.001	1E-20	0.01	1E-6	0.02	400	1E-5
Rössler z	0.06	0.5	1E-12	0.04	1E-8	0.1	10	1E-6
DST 2013	3E-4	0.3	1E-18	0.003	1E-6	0.04	2	1E-5
DST 2014	4E-4	0.35	1E-18	0.003	1E-6	0.08	4	1E-6

In addition, three DFRs were evaluated: a general DFR that uses the MG model as the nonlinear element (MG DFR), the proposed modular DFR that uses an identity function as f (identity DFR) and the proposed modular DFR that uses a hyperbolic tangent function as f (tanh DFR). Python was used to obtain the prediction accuracy of the DFRs. The mask signal had two values, +1 and −1, and the number of virtual nodes N_x was 100 for all reservoirs. In the MG DFR, fixed parameters $p = 1$ and $\theta = 0.2$ were used for all datasets. For the identity and tanh DFRs, we used $B = 0.82$. Here, $\theta = 0.2$ is the value at which the DFR is considered to work effectively because coupling between the virtual nodes is optimal [2]. Because $B = e^{-\theta}$, fixing $\theta = 0.2$ was equivalent to fixing $B = 0.82$. Other parameters were optimized by grid search, and the values used are summarized in Table 1. Note that the weights of the linear transform in the output layer were learned by ridge regression using the regularization parameter, β. For each dataset, normalization was implemented with the maximum value set to 1. For the identity and tanh DFRs, β needed to be multiplied by $(1/\max_k u(k))^2$.

The results of the accuracy evaluation on the Lorentz time series are summarized in Table 2. To evaluate the accuracy of the proposed methods and compare them with existing ones, we used the root mean square error (RMSE) and the mean absolute error (MAE), as in [22]. The RMSE and MAE were defined based on the predicted output (y_i), target output (d_i), and number of samples in the test data (n) as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (d_i - y_i)^2}, \tag{27}$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |d_i - y_i|. \tag{28}$$

The accuracy evaluation results for the Rössler and DST time series are presented in Tables 3 and 4, respectively.

Figure 9 plots the logarithm of the reciprocal of the ratio of the RMSE of each DFR methods to those of MOICBLS, the state-of-the-art method for all above datasets. It shows that the further the bar extends to the right, the better is the method. In the figure, the proposed tanh DFR exhibits excellent time-series forecasting accuracy even more than the MG DFR. We also obeserve that the identity DFR is as accurate as an MG DFR and a tanh DFR for most datasets. For some datasets, such as Lorenz y and Lorenz z, nonlinearity of f is considered necessary to achieve high accuracy. For future comparisons, evaluation results on additional datasets are presented in the Appendix.

Table 2. Prediction results of Lorenz time series. Values except for DFRs are obtained from [22]. The best and second-best values for each dataset are shown in bold and underlined, respectively.

methods	RMSE			MAE		
	x	y	z	x	y	z
RF	3.504E-4	5.265E-4	7.820E-4	3.363E-4	6.028E-4	6.890E-4
ELM	9.113E-4	2.226E-3	1.968E-3	8.746E-4	2.515E-3	1.768E-3
SVM	4.525E-3	5.774E-3	5.553E-2	4.342E-3	6.140E-3	5.471E-2
Elman	2.867E-4	5.481E-4	6.746E-3	2.752E-4	5.679E-4	6.402E-4
DAE	1.017E-3	2.384E-3	2.013E-3	9.765E-4	2.686E-3	1.994E-3
GB	3.237E-4	5.607E-4	7.117E-4	3.106E-4	6.916E-4	6.855E-4
ESN	3.122E-4	5.567E-4	6.716E-4	2.996E-4	6.023E-4	6.675E-4
BLS	2.884E-4	5.277E-4	6.299E-4	2.859E-4	5.676E-4	6.245E-4
MOICBLS	2.591E-4	4.089E-4	**4.766E-4**	2.486E-4	4.424E-4	<u>4.523E-4</u>
MG DFR	<u>7.996E-5</u>	3.219E-4	7.208E-4	<u>5.337E-5</u>	2.200E-4	5.200E-4
identity DFR	1.353E-4	3.195E-3	3.986E-3	9.059E-5	2.344E-3	2.645E-3
tanh DFR	**2.916E-5**	**2.888E-4**	<u>6.281E-4</u>	**2.078E-5**	**1.786E-4**	**4.358E-4**

Table 3. Prediction results of Rössler time series. Values except for DFRs are obtained from [22]. The best and second-best values for each dataset are shown in bold and underlined, respectively.

methods	RMSE			MAE		
	x	y	z	x	y	z
RF	8.158E-4	7.665E-3	3.910E-3	8.363E-4	7.256E-3	3.445E-3
ELM	1.262E-3	1.096E-3	9.844E-3	1.139E-3	1.209E-3	8.846E-3
SVM	6.215E-3	9.075E-3	2.776E-1	5.756E-3	8.463E-3	2.735E-1
Elman	3.867E-3	6.848E-4	3.373E-2	2.944E-3	5.679E-4	3.201E-3
DAE	3.502E-3	7.758E-4	1.006E-2	3.765E-3	7.349E-4	9.972E-3
GB	6.237E-4	8.690E-4	3.558E-3	6.106E-4	8.619E-4	3.427E-3
ESN	5.688E-4	2.517E-4	3.357E-2	5.037E-4	2.319E-4	3.337E-3
BLS	5.335E-4	2.327E-4	3.149E-2	4.725E-4	2.162E-4	3.111E-3
MOICBLS	3.559E-4	1.210E-4	2.383E-3	3.239E-4	1.115E-4	2.226E-3
MG DFR	2.256E-5	1.443E-5	**1.391E-4**	1.100E-5	1.222E-5	**9.923E-5**
identity DFR	<u>6.069E-6</u>	<u>3.851E-6</u>	6.940E-4	<u>1.980E-6</u>	<u>3.228E-6</u>	4.241E-4
tanh DFR	**6.067E-6**	**2.841E-6**	<u>2.174E-4</u>	**1.724E-6**	**1.915E-6**	<u>1.080E-4</u>

4.2 Comparison with existing hardware implementations

The hardware implementation of the proposed identity DFR model, which performs well on most datasets and is the easier of the two proposed DFRs to implement, was evaluated.

The proposed identity DFR model (Prop.) was implemented in an FPGA board using Zynq-7000 (xc7z020clg400-1). The detailed implementation of the identity DFR is illustrated in Fig. 10. The masking procedure was implemented with a sign inverter, a multiplexer, and shift registers that store 1-bit weight. The mask signal adopted two values, +1 and −1. The mask processing was applied to one input signal $u(k)$, resulting in N_x reservoir input signals $j(k)_n$. In the following experiments, the number of virtual nodes in the reservoir, N_x, was set as 100. An approximate multiplication using the sum of three right shifts with different shifts was adopted for acceleration

Table 4. Prediction results of DST time series. Values except for DFRs are obtained from [22]. The best and second-best values for each dataset are shown in bold and underlined, respectively.

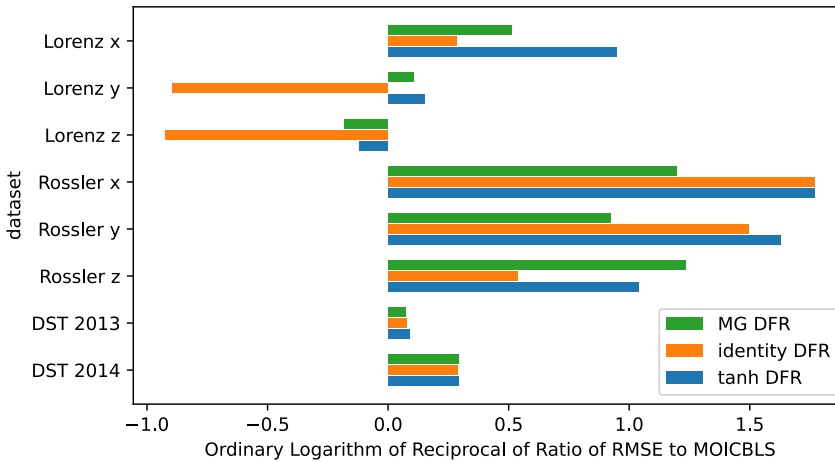| methods	DST 2013		DST 2014	
	RMSE	MAE	RMSE	MAE
RF	5.660	5.313	12.669	11.217
ELM	3.866	3.965	8.529	7.988
SVM	3.815	3.802	8.059	6.642
Elman	4.006	4.028	8.632	7.798
DAE	3.805	3.798	8.438	7.911
GB	4.652	4.293	10.315	9.319
ESN	3.985	3.905	8.648	7.912
BLS	3.972	3.885	8.471	7.854
MOICBLS	3.456	3.444	7.663	6.923
MG DFR	2.923	1.951	<u>3.927</u>	<u>2.898</u>
identity DFR	<u>2.898</u>	<u>1.946</u>	3.964	2.905
tanh DFR	**2.818**	**1.930**	**3.914**	**2.886**

Fig. 9. Logarithm of reciprocal of ratio of RMSE of DFR method to that of MOICBLS. A bar extending further to the right represents a superior method to MOICBLS, whereas positive or negative values indicate whether a method is better or worse, respectively, than MOICBLS.

and resource usage-saving. A and B in Fig. 10 correspond to $\eta(1-e^{-\theta})$ and $e^{-\theta}$ in Fig. 4, respectively. The feedback in the reservoir was configured by the shift registers. Here, for each input signal $u(k)$, N_x reservoir features $x(k)_n$ were obtained. Linear transformation was implemented with a multiplier, an adder, and shift registers storing the weight. Each $x(k)_n$ was multiplied by the corresponding weight, w_n, and summed, followed by addition of a constant term to obtain the output, $y(k)$.

Fixed-point number representation was used throughout the experiments. Different bit widths were used for the reservoir and output layers. When the number of bits in the reservoir layer was r, the input, $u(k)$, was normalized; therefore, its maximum value was $l = 2^{r-2}$. Similarly, when the number of bits in the output layer was o, the target output, $d(k)$, was normalized such that its
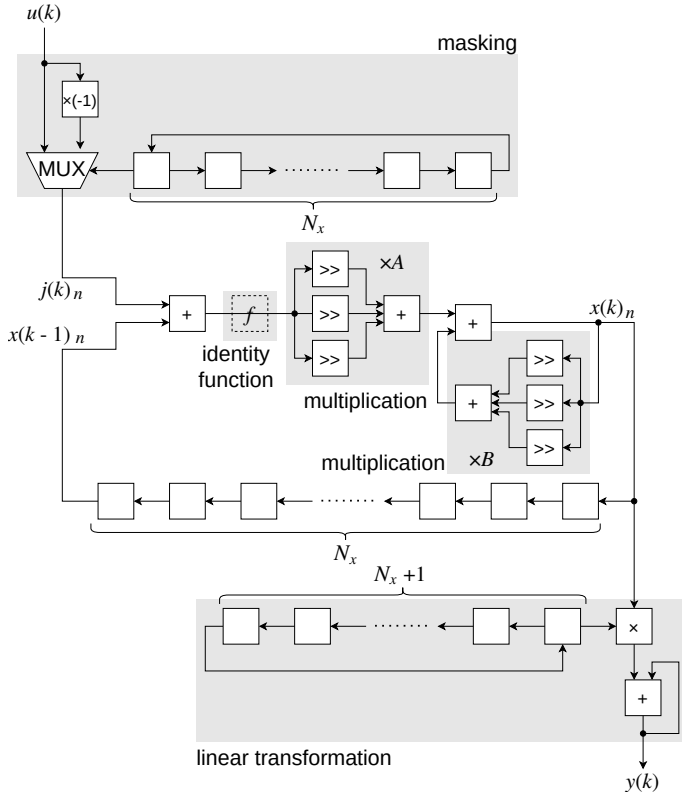
Fig. 10. Implementation of the proposed identity DFR. Blank blocks with cyclic arrow connection form a shift register. Shift for each approximation multiplier block is optimized.

maximum value was $m = 2^{o-1}$. The weights of the output layer were learned offline using Python. The C++ code was synthesized in Xilinx Vitis HLS 2021.1, and the circuit was implemented and evaluated in Xilinx Vivado 2021.1.

The proposed identity DFR was compared with state-of-the-art analog and digital implementations. The analog implementation compared was a hybrid FPGA–ASIC DFR (hybrid DFR) [16]. This hybrid DFR employed a nonlinear element based on the MG model, which was realized as an analog circuit on a 180 nm ASIC. The remaining circuits, including the ADC, were implemented using the those on the same FPGA as in the proposed method, whereas a discrete DAC component was used. In the following evaluations, we adopted the same two tasks as in [16], namely, NARMA10 and spectrum sensing. The algorithm of the proposed identity DFR is different from those of MG-based DFRs (MG DFR and hybrid DFR), and simplifications such as approximate arithmetic operation and bitwidth reduction were employed in the identity DFR. Therefore, the accuracy of these tasks should be different from that when using the compared methods.

The digital implementation compared was [2] (naive DFR), in which a DFR using the MG model is implemented using a 32-bit floating point number representation. This corresponds to the implementation discussed in Section 2.4. The input and output layers are the same as those used in the proposed method, with N_x=100. None of the above implementations use pipelining.

Table 5. Parameters of DFRs used for the comparison of hardware implementations. The A value represents the amount of shift in the three right shifts that constitute multiplication by A.

dataset	Naive DFR			Prop.	
	γ	η	β	A	β
NARMA10	0.05	0.5	1E-16	{4,5,11}	4E-12
Spectrum Sensing	0.01	0.01	1E-3	{7,9,11}	1E+4

Table 6. Bit width configurations and accuracy comparison on the NARMA10 task. The NRMSE of hybrid DFR is obtained from [16].

Method	Naive DFR	Hybrid DFR	Prop. (16-bit)	Prop. (24-bit)
Nonlinear Element	Mackey-Glass	Mackey-Glass	Not used (Identity)	Not used (Identity)
Number rep. in the reservoir	32 bit floating	16 bit fixed	16 bit fixed	24 bit fixed
Number rep. in the output layer	32 bit floating	32 bit fixed	21 bit fixed	29 bit fixed
NRMSE	0.14	0.21	0.20	0.14

In the naive DFR, fixed parameters of $p = 1$ and $\theta = 0.2$ were employed for both datasets. For the identity DFR, multiplication by B was approximated using the summation of the three right shifts, $\{1, 2, 4\}$. Other parameters were optimized by grid search. The values of the parameters used are summarized in Table 5. Note that the indicated values are applicable before normalization of the input, $u(k)$. For the identity DFR, β must be multiplied by $(l/\max_k u(k))^2$ based on Eq. (22).

The results of the accuracy evaluation on NARMA10 are summarized in Table 6. Here, the accuracy is compared in terms of normalized RMSE (NRMSE), as in [16].

$$\text{NRMSE} = \frac{||\boldsymbol{y} - \boldsymbol{d}||}{||\boldsymbol{y}||}, \tag{29}$$

where $\boldsymbol{y}$ is the vector of all outputs in the test data, and $\boldsymbol{d}$ is the vector of the target outputs. $|| \cdot ||$ is the Euclidean norm. The proposed identity DFR achieved high accuracy, even when nonlinear elements were substituted with the identity function. When the 24-bit fixed-point representation was used, the accuracy of the identity DFR was comparable to that of the naive DFR using the standard floating-point number representation.

We then compared the accuracy using spectrum sensing (Table 7). Here, the area under the curve (AUC) was evaluated. The proposed identity DFR (16-bit) achieved equal or better accuracy than the existing implementations under all conditions in this task. When the reservoir used a 16-bit precision, the accuracy was equivalent to a software implementation using floating point numbers.

Subsequently, we evaluated the circuit size, power consumption, and throughput. Owing to space limitations, only the evaluation results on NARMA10 are presented. In these comparisons, the operation frequency has a deterministic impact. The reported operation frequency of the hybrid DFR is 10 MHz [16], probably owing to the use of a DAC and an ADC. Higher frequencies correspondingly increase the power consumption of the DAC. Conversely, fully digital implementations, namely, the naive and identity DFRs, operate at significantly higher frequencies, such as 100 MHz or higher. Consequently, we compared these performances separately with those of existing analog and digital implementations.

Table 7. AUC comparison (spectrum sensing). The values of the hybrid DFR are obtained from [16]. Bold font indicates the best result.

SNR (dB)	Antennae count	Naive DFR	Hybrid DFR	Prop. (16-bit)
−10	2	**0.920**	0.913	**0.920**
−10	4	**0.995**	0.994	**0.995**
−10	6	**1.000**	0.999	**1.000**
−15	2	0.775	0.770	**0.776**
−15	4	0.937	0.934	**0.937**
−15	6	**0.989**	0.988	**0.989**
−20	2	**0.630**	0.603	**0.630**
−20	4	**0.764**	**0.764**	**0.764**
−20	6	**0.871**	**0.871**	**0.871**

Table 8. Comparison of circuit size and power consumption (NARMA10). Bit widths are the same as those in Table 6.

No. of DFR	Hybrid DFR	Prop. (16-bit)
Slice LUTs	270	312
Slice Registers	407	374
Slice	134	126
Block RAM Tile	2	1.5
DSPs	3	1
Power in digital domain	1 mW	1 mW
Power in analog domain	> 10 mW	−

4.2.1 Comparison with analog implementation. Table 8 compares the circuit size and power consumption with those of the hybrid DFR. Here, the number of logic elements and power consumption for the digital circuits in the DFR, excluding the DAC, analog circuit, and ADC, was calculated by implementing each method on the same FPGA platform at an operating frequency of 10 MHz. Conversely, for the hybrid DFR, the power consumption of the ADC, DAC, and ASIC were needed to be added to the power of the logic circuits. The reported power consumption of the ASIC, which provides a nonlinear function, was 24.55 µW [16]. The hybrid DFR employed a 16-bit DAC with a reference voltage of 2.5 V and a Zynq-7000 embedded 12-bit XADC with a reference voltage of 1 V. The maximum power consumption of the XADC was 36 mW [19]. A key metric to assess converter performance is the figure of merit (FoM) [15], defined by the following equation [15]:

$$\text{FoM} = \frac{P}{f_\text{s} \cdot 2^\text{ENOB}}, \tag{30}$$

where P is the power, f_s is the Nyquist sampling rate, and

$$\text{ENOB} = \frac{\text{SNDR(dB)} - 1.76}{6.02}, \tag{31}$$

where SNDR is the signal-to-noise-and-distortion ratio. We estimated that the power of this ADC alone was 10 mW[1]. When considering the power usage of other components, including the discrete

---

[1]The lowest limit of the FoMs for ADCs reported in [15] was 100 fJ/(conv-step). Considering the specifications of the XADC (sampling frequency f_s=10 MHz and effective number of bits ENOB=12), the lowest possible power consumption was 4 mW when FoM=100 fJ/(conv-step). It was estimated to consume 10 mW for a more realistic value of FoM=250 fJ/(conv-step).

Table 9. Comparison of circuit size and power consumption (NARMA10). Bit widths are the same as those in Table 6.

No. of DFR	Naive DFR	Prop. (16-bit)
Slice LUTs	5412	312
Slice Registers	7406	374
Slice	1983	131
Block RAM Tile	3	1.5
DSPs	27	1
Power	58 mW	7 mW

Table 10. Throughput comparison. Value of hybrid DFR is obtained from [16].

Method	Hybrid DFR	Naive DFR	Prop. (16-bit)
Clock Frequency (MHz)	10	100	100
Throughput (samples/s)	1625	4670	8608

DAC, the power consumption of the proposed identity DFR was at least 10 times smaller than that of the hybrid DFR.

4.2.2 Comparison with digital implementation. Table 9 compares the circuit size and power consumption with those of a naive DFR. Both DFRs were implemented using the same FPGA and operated at a frequency of 100 MHz. According to the results, the proposed identity DFR is 8.3 times more power efficient than the naive DFR implementing the MG model.

4.2.3 Comparison of the throughput. Table 10 presents the results of the throughput evaluation. Based on the table, the proposed identity DFR achieves 5.3 and 1.8 times higher throughput than the hybrid implementation and naive DFR, respectively.

4.2.4 Discussion. This discussion focuses on the effectiveness of the proposed identity DFR with respect to two specific tasks in [16]. Although the identity DFR was proven be effective for these tasks, its accuracy slightly lagged that of the DFR using the MG model for certain other tasks. Therefore, for these tasks, it may be necessary to introduce nonlinearity in f. For example, using the MG model as f provides nonlinearity with tunable parameters. However, even in such cases, the modular DFR model proposed in Section 3.1 can provide a lighter implementation option and facilitate a power− and performance−efficient design suitable for embedded processing.

5 CONCLUSION

This paper proposed a new DFR model, called modular DFR, suitable for efficient digital implementation. This new model provides greater flexibility in selecting the nonlinear structure without increasing the complexity of the design. As practical examples of the proposed DFR model, identity and tanh DFRs were presented, which use identity and hyperbolic tangent functions, respectively, as the pluggable nonlinear function of the proposed modular DFR model. The software simulation results demonstrated that the proposed tanh DFR outperforms state-of-the-art chaotic time series prediction methods. The identity DFR also shows comparable accuracy to the tanh DFR on most datasets. Moreover, the implementation using an FPGA showed efficiency of the proposed identity DFR. It achieves 10x and 8.3x reduction in power consumption, and a 5.3x and 1.8x increase in throughput compared to those of state-of-the-art analog and digital implementations, respectively.

ACKNOWLEDGMENTS

REFERENCES

[1] Miquel L Alomar, Miguel C Soriano, Miguel Escalona-Morán, Vincent Canals, Ingo Fischer, Claudio R Mirasso, and Jose L Rosselló. 2015. Digital implementation of a single dynamical node reservoir computer. *IEEE Transactions on Circuits and Systems II: Express Briefs* 62, 10 (2015), 977–981.

[2] Lennert Appeltant, Miguel Cornelles Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, Joni Dambre, Benjamin Schrauwen, Claudio R Mirasso, and Ingo Fischer. 2011. Information processing using a single dynamical node as complex system. *Nature Communications* 2, 1 (2011), 1–6.

[3] Lennert Appeltant, Guy Van der Sande, Jan Danckaert, and Ingo Fischer. 2014. Constructing optimized binary masks for reservoir computing with delay systems. *Scientific Reports* 4, 1 (2014), 1–5.

[4] Adrija Banerjee, Amaresh Bej, and TN Chatterjee. 2012. On the existence of a long range correlation in the geomagnetic disturbance storm time (Dst) index. *Astrophysics and Space Science* 337 (2012), 23–32.

[5] Rohitash Chandra. 2015. Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction. *IEEE Transactions on Neural Networks and Learning Systems* 26, 12 (2015), 3123–3136.

[6] CL Philip Chen and Zhulin Liu. 2017. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Transactions on Neural Networks and Learning Systems* 29, 1 (2017), 10–24.

[7] Thao-Tsen Chen and Shie-Jue Lee. 2015. A weighted LS-SVM based learning system for time series forecasting. *Information Sciences* 299 (2015), 99–116.

[8] Data Analysis Center for Geomagnetism and Space Magnetism. 2022. Plot and data output of Dst and AE indices (Hourly Values). https://wdc.kugi.kyoto-u.ac.jp/dstae/index.html.

[9] Pei Guo, Chen Liuy, Yan Tang, and Jianwu Wang. 2019. Parallel gradient boosting based granger causality learning. In *IEEE International Conference on Big Data*. IEEE, 2845–2854.

[10] Min Han and Meiling Xu. 2017. Laplacian echo state network for multivariate time series prediction. *IEEE Transactions on Neural Networks and Learning Systems* 29, 1 (2017), 238–244.

[11] Sosei Ikeda, Hiromitsu Awano, and Takashi Sato. 2022. Hardware-friendly delayed-feedback reservoir for multivariate time-series classification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 3650–3660. https://doi.org/10.1109/TCAD.2022.3197488

[12] Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. 2002. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation* 10, 3 (2002), 263–282.

[13] Mantas Lukoševičius and Herbert Jaeger. 2009. Reservoir computing approaches to recurrent neural network training. *Computer Science Review* 3, 3 (2009), 127–149.

[14] Michael C Mackey and Leon Glass. 1977. Oscillation and chaos in physiological control systems. *Science* 197, 4300 (1977), 287–289.

[15] Boris Murmann. 2008. A/D converter trends: Power dissipation, scaling and digitally assisted architectures. In *IEEE Custom Integrated Circuits Conference*. 105–112.

[16] Osaze Shears, Kangjun Bai, Lingjia Liu, and Yang Yi. 2021. A Hybrid FPGA-ASIC delayed feedback reservoir system to enable spectrum sensing/sharing for low power IoT devices. In *IEEE/ACM International Conference on Computer Aided Design*. 1–9.

[17] Miguel C Soriano, Silvia Ortín, Lars Keuninckx, Lennert Appeltant, Jan Danckaert, Luis Pesquera, and Guy Van der Sande. 2014. Delay-based reservoir computing: Noise effects in a combined analog and digital implementation. *IEEE Transactions on Neural Networks and Learning Systems* 26, 2 (2014), 388–393.

[18] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. 2019. Recent advances in physical reservoir computing: A review. *Neural Networks* 115 (2019), 100–123.

[19] Xilinx. 2022. 7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide (UG480). https://docs.xilinx.com/r/en-US/ug480_7Series_XADC.

[20] Yaxu Xue, Xiaofei Ji, Dalin Zhou, Jing Li, and Zhaojie Ju. 2019. SEMG-based human in-hand motion recognition using nonlinear time series analysis and random forest. *IEEE Access* 7 (2019), 176448–176457.

[21] Ke Yan, Zhiwei Ji, Huijuan Lu, Jing Huang, Wen Shen, and Yu Xue. 2017. Fast and accurate classification of time series data using extended ELM: Application in fault diagnosis of air handling units. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49, 7 (2017), 1349–1356.

[22] Jun Yi, Jiahua Huang, Wei Zhou, Guorong Chen, and Meng Zhao. 2022. Intergroup cascade broad learning system with optimized parameters for chaotic time series prediction. *IEEE Transactions on Artificial Intelligence* 3, 5 (2022),

Table 11. Parameters of DFRs used for additional datasets.

dataset	MG DFR			identity DFR		tanh DFR		
	γ	η	β	A	β	A	α	β
DST 2021	4E-4	0.4	1E-18	0.004	1E-6	0.03	2	1E-5
DST 2022	5E-4	0.4	1E-18	0.002	1E-5	0.09	2	1E-5
Chen_x	0.03	0.3	1E-12	0.03	1E-7	0.07	7	1E-5

Table 12. Prediction results of DFRs on additional datasets.

methods	DST 2021		DST 2022		Chen_x	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
MG DFR	3.141	2.116	3.782	2.784	1.259E-4	9.169E-5
identity DFR	3.171	2.123	3.841	2.831	7.851E-4	4.783E-4
tanh DFR	3.146	2.117	3.792	2.773	1.156E-4	6.158E-5

709–721. https://doi.org/10.1109/TAI.2022.3143079

A ADDITIONAL EVALUATION FOR FUTURE COMPARISONS

This section presents the evaluation results on additional datasets for future comparisons.

The datasets used comprised the most recent 2021 and 2022 DST datasets, along with the Chen system, a typical chaotic system. For the DST datasets, 4,000 samples were selected from the tails of both the 2021 and 2022 data. The initial 3,000 samples were employed for training, whereas the remaining 1,000 samples were reserved for testing, following the methodology described in Section 4.

The Chen system is defined as:

$$
\begin{aligned}
\frac{dx}{dt} &= a(y - x), \\
\frac{dy}{dt} &= (c - a)x - xz + cy, \\
\frac{dz}{dt} &= xy - bz,
\end{aligned}
\tag{32}
$$

where $a = 40$, $b = 3$, $c = 28$, $x(0) = -0.1$, $y(0) = 0.5$, and $z(0) = -0.6$ in the following experiments. The fourth-order Runge–Kutta method was used to generate 12,000 samples with a sampling interval of 0.01 s. Among these, the first 9,000 samples were used for training, and the remaining 3,000 samples were used for testing. The DFRs were trained to predict the value of x.

The evaluation conditions were the same as in Section 4, and the parameter values used are summarized in Table 11.

The accuracy evaluation results are summarized in Table 12.

The forecast result for the Chen system using the proposed tanh DFR is shown in Figure 11. The upper figure compares the predicted and true values, and the lower figure plots the absolute error.
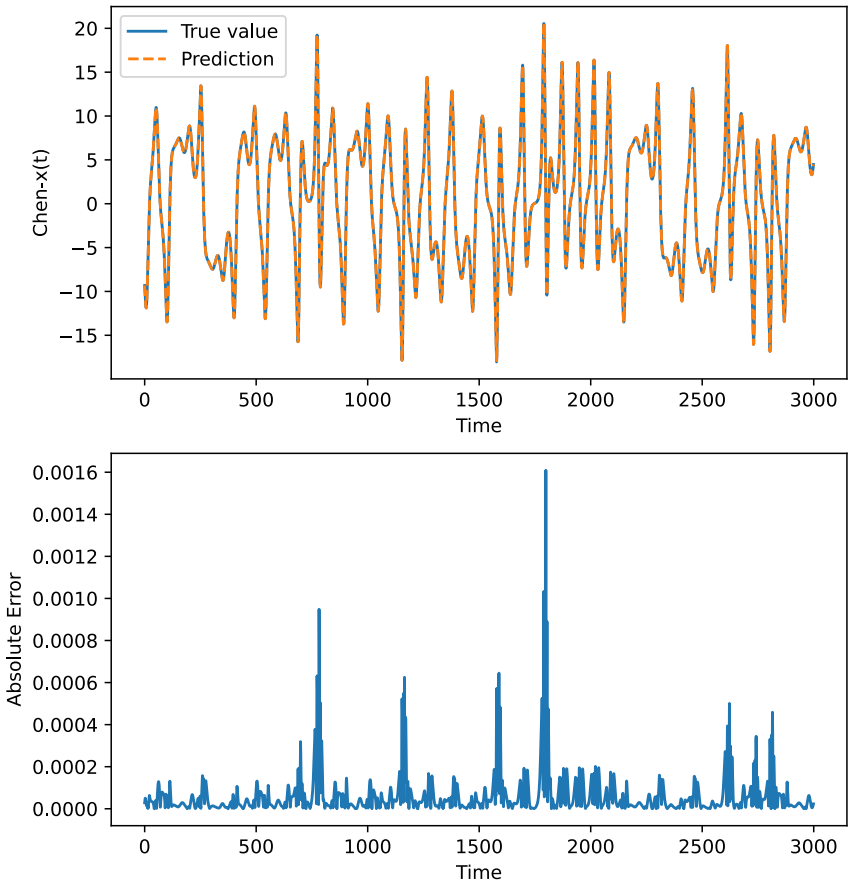
Fig. 11. Comparison of true and predicted values and absolute error by tanh DFR for the Chen system.