Digital Implementation of a Single Dynamical Node Reservoir Computer

Miquel L. Alomar, Miguel C. Soriano, *Member, IEEE*, Miguel Escalona-Morán, Vincent Canals, *Member, IEEE*, Ingo Fischer, Claudio R. Mirasso, and Jose L. Rosselló, *Member, IEEE*

***Abstract*—Minimal hardware implementations of machine-learning techniques have been attracting increasing interest over the last decades. In particular, field-programmable gate array (FPGA) implementations of neural networks (NNs) are among the most appealing ones, given the match between system requirements and FPGA properties, namely, parallelism and adaptation. Here, we present an FPGA implementation of a conceptually simplified version of a recurrent NN based on a single dynamical node subject to delayed feedback. We show that this configuration is capable of successfully performing simple real-time temporal pattern classification and chaotic time-series prediction.**

***Index Terms*—Artificial neural networks (ANNs), field-programmable gate arrays (FPGAs), hardware (HW), multiple signal classification, neural network (NN), pattern recognition, recurrent neural networks (RNN), time-series prediction.**

I. Introduction

OUR modern society is generating increasingly large amounts of information, often even referred to as the age of big data. Machine learning, particularly recurrent neural network (RNN) approaches, represents one of the most successful attempts to process large amounts of sequential information in a meaningful manner.

Reservoir computing (RC) stands out as a relatively simple technique to implement RNNs [1]–[3]. Unlike Hopfield networks where the connection weights between network nodes have to be trained for specific tasks, in RC, weights are randomly initialized and left fixed. The training procedure is only applied to the connection weights between the recurrent network and its output layer that is used for, for example, classification or other applications. This training procedure enormously simplifies the implementation and can be performed, in practice, via a simple linear regression.

While RC is usually realized in software, this concept is particularly appealing for hardware implementations since the connections in the recurrent network do not need to be changed in time. For increasing convenience of hardware implementations, it has been recently shown that a single nonlinear node with self-feedback can replace the recurrent network on the reservoir [4]. The main advantages of delay-based systems are their conceptual simplicity and potentially low hardware requirements [5]–[8] resulting from the possible implementation within a RAM memory. The high regularity and area efficiency of these memories facilitate to compact the complex reservoir networks.

Here, we present a self-contained digital implementation of the single-node reservoir computer with delayed feedback. The final solution is hosted within a field-programmable gate array (FPGA), with the delay line being implemented using a RAM memory. The main advantage of using an FPGA solution with respect to the standard microprocessor-based alternative is the possibility of implementing both, the arithmetic units and the reservoir, within the same chip. The utilization of an on-chip memory to implement the reservoir network enables a higher throughput and implies lower power consumption than using onboard memories (the memory access speed of an onboard memory is on the order of gigabytes per second, whereas that of an on-chip one is on the order of terabytes per second). Therefore, the use of a compact implementation for the whole system in a single integrated circuit is interesting from the energy efficiency point of view. It can be a solution for those electronic systems implementing computational intelligence techniques and requiring low power dissipation (such as wireless sensor networks [9]). Importantly, the replacement of the recurrent network, with a large number of nodes, by a single network node subject to delayed feedback relaxes the hardware requirements for the FPGA. To demonstrate the capabilities of such a device, we evaluate its performance in a simple pattern recognition task and a benchmark time-series prediction task.

FPGA-based realizations of artificial NN (ANN) models are numerous [10]. Nonetheless, chip implementations of the RC concept are scarce [11], [12]. The present work is the first digital implementation of the RC approach using a single nonlinear oscillator with delayed feedback as dynamical node.

II. Methods

In traditional RC, an input signal is injected from the input layer to the reservoir via random connections. The large number of nodes in the reservoir creates a high-dimensional

Manuscript received March 17, 2015; accepted May 19, 2015. Date of publication July 17, 2015; date of current version September 25, 2015. This work was supported in part by the Spanish Ministry of Economy and Competitiveness (MINECO), the Regional European Development Funds (FEDER), and the Comunitat Autònoma de les Illes Balears under Grant contracts TEC2011-23113, TEC2012-36335, and TEC2014-56244-R, Grups Competitius and a fellowship (FPI/1513/2012) financed by the European Social Fund (ESF) and the Govern de les Illes Balears (Conselleria d'Educació, Cultura i Universitats). This brief was recommended by Associate Editor G. Masera.

M. L. Alomar, V. Canals, and J. L. Rosselló are with the Electronics Engineering Group, Physics Department, Universitat de les Illes Balears, 07122 Palma de Mallorca, Spain (e-mail: miquellleo.alomar@uib.es; vincent.canals@gmail.com; j.rossello@uib.es).

M. C. Soriano, M. Escalona-Morán, I. Fischer, and C. R. Mirasso are with Instituto de Física Interdisciplinar y Sistemas Complejos, IFISC (CSIC-UIB), Universitat de les Illes Balears, 07122 Palma de Mallorca, Spain (e-mail: miguel@ifisc.uib-csic.es; miguelangel@ifisc.uib-csic.es; ingo@ifisc.uib-csic.es; claudio@ifisc.uib-csic.es).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2015.2458071

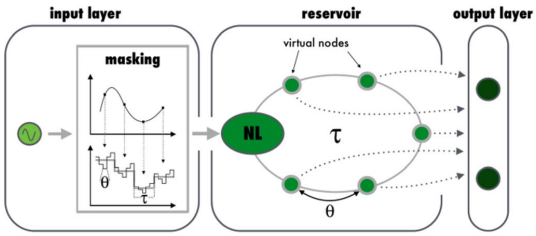

Fig. 1. Schematic view of a reservoir computer based on a single nonlinear node (NL) with delay (τ). Virtual nodes are defined as temporal positions in the delay line.

space (much higher than the dimension of the input signal) [3]. The effect of the input signals onto the nodes in the recurrent network is to generate transient dynamical states that are later read out and classified in the output layer. For the training of the classifiers, each input is assigned a corresponding target value. Our reservoir computer is based on the emulation of a recurrent network via a single nonlinear dynamical node subject to delayed feedback [4]. Fig. 1 depicts the general concept of this approach, including the three layers required for RC: the input layer, the delay-based reservoir, and the output layer. In the reservoir, we define N equidistant points separated in time by $\theta = \tau/N$ within one delay interval of length τ. We denote these N equidistant points as virtual nodes," as they play a role analogous to the nodes in a traditional reservoir. The values of the delayed variable at each of the N points define the states of the virtual nodes. These states characterize the transient response of the nonlinear node (NL) to a certain input at a given time. The separation time θ among virtual nodes is chosen to optimize the reservoir performance. We choose $\theta < T$, with T being the characteristic time scale of the nonlinear node. Via this choice, the states of the virtual nodes become dependent on the states of the neighboring nodes. Interconnected in this way, the virtual nodes emulate a network serving as reservoir.

The virtual nodes are subjected to the time-continuous or discrete input stream which can be a time-varying scalar variable or vector of any dimension. The feeding to the individual virtual nodes is achieved by time multiplexing the input signal. Fig. 1 illustrates the input layer preprocessing for a scalar input. For this, the input stream undergoes a sample and hold operation to define a stream that is constant during one delay interval τ, before it is updated with the next sample in the input signal. To emulate the random weights from the input layer to the reservoir in traditional RC, we introduce a random matrix mask. Upon carrying out the multiplication of the mask with the sample at a certain time t_0 of the input signal, we obtain an N-dimensional vector which represents the temporal input sequence within the interval $[t_0,\ t_0 + \tau)$ [4]. Each virtual node is updated every time τ. After processing the input signal, a training algorithm assigns an output weight to each virtual node, such that the weighted sum of the states approximates the desired target value as closely as possible. The training follows the standard procedure for RC [13], [14]. The testing is then performed using previously unseen input data of the same kind of those used for training.

Fig. 2 schematically shows the digital implementation of the proposed reservoir computer. The core of the digital implementation is a differential equation solver that emulates the Mackey–Glass oscillator [15] as the nonlinear node. This oscillator together with the RAM block represents the delay-based reservoir depicted in Fig. 1. The Mackey–Glass node receives the external input, after masking the signal in the input layer [4], in our case with 8-bit resolution. The precision of the digital representation of the input and output signals has a major impact on the ultimate performance of the system [8]. In the global implementation, a control block configures both the dynamical system with the desired parameters as well as the external memory with the mask and weight values for each node and output class. In addition, it states which memory value should be provided at each time step. An external C++ program specifies to the control block whether the system must enter into a configuration mode (in which it is arranged with the proper parameters and memory values) or into the operation mode in order to process the input signals. The configuration parameters, mask, weights, and input values are all stored in external data files. The orders to choose the system's mode and all of the data are sent to the system by the C++ program via the serial port.

Once the system has been configured, it is provided with an input value every delay interval τ, as explained before. The input signal is time multiplexed within each τ by multiplying each mask value in time steps $\theta = \tau/N$ before feeding the dynamical system. The values of the delayed variable x at each time $j\theta$ (individual virtual nodes' outputs with $j = 1, \ldots, N$) are stored in the internal memory of the FPGA and supplied to the classification block. Finally, the output layer performs every time θ a multiplication of the state x_j by the corresponding weight ω_{jl} for each one of the c output classes ($l = 1, \ldots, c$). Only one class needs to be considered ($c = 1$) when the system performs a prediction task.

To perform temporal classification, the result of this product is sequentially added for a given number (α) of intervals τ

$$y_l = \sum_{j=1}^{\alpha \cdot N} \omega_{jl} \cdot x_j. \tag{1}$$

The greatest value of the weighted sum of the states y_k determines the output category that matches the input signal, i.e., our system follows a winner-take-all strategy. In the case of time-series prediction, the weighted sum (1) is computed with $\alpha = 1$, providing a predicted value at each interval τ.

The numerical quantities are represented in a digital format adopting the fixed-point notation. In particular, unsigned notation is used to represent all variables except the output weights, which require an additional sign bit. A number of 8 fractional bits and no integer bits are used for the input signal as well as for the mask values and the delayed variable x. Consequently, their working range is limited to the [0, 1) interval. Intermediate variables are given a greater resolution when necessary. The output weights ω_{jl} and the final classification outputs y_l are provided with a 16-bit resolution.

The differential equation solver shown in Fig. 2 is implemented using digital logic. The digital solver reproduces the differential equation

$$\dot{x} = (k - x)\frac{1}{T} \tag{2}$$

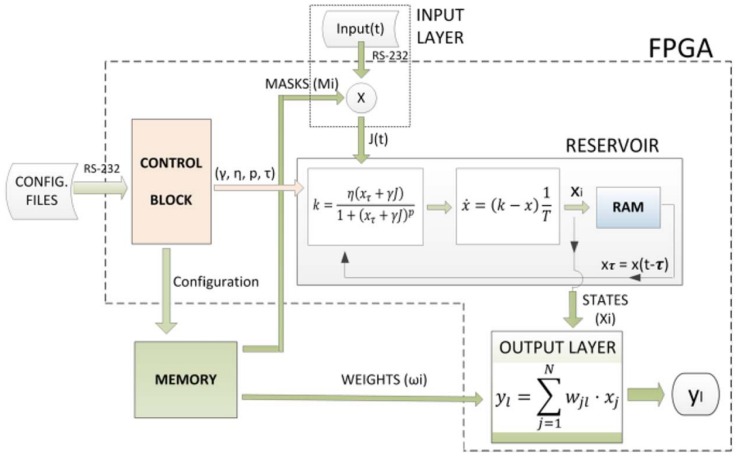

Fig. 2. Diagram of the reservoir computer implemented in the FPGA.

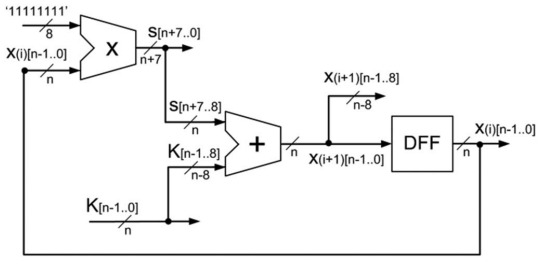

Fig. 3. Schematic of the differential equation solver. First-order Euler method is implemented by means of an adder, a multiplier, and a register. 16 bits ($n = 16$) are used within the solver.

where T is the characteristic time of the system. Expression (2) is reduced to the Mackey–Glass oscillator when k takes the form

$$k = \frac{\eta(x_\tau + \gamma J)}{1 + (x_\tau + \gamma J)^p} \tag{3}$$

where the variable k is implemented by an appropriate digital arithmetic block and x_τ is the delayed version of x provided by the RAM memory. Using a time step fixed to $\Delta t = T/255$ and assuming that $\dot{x} \simeq (x_{i+1} - x_i)/\Delta t$ (first-order Euler method), the differential equation (2) can be replaced by the next recurrent expression

$$x_{i+1} = x_i + \frac{k - x_i}{255}. \tag{4}$$

Expression (4) is implemented digitally by using only an adder, a multiplier, and D flip-flops as it is shown in Fig. 3. Even though an amount of 16 bits ($n = 16$) is used to codify the variable x within the differential equation solver, the output (x_{i+1}) is provided to the RAM memory with 11-bit resolution and to the classification block with 8-bit resolution. The multiplication of the current state value x_i by 255 is performed in order to obtain the value of s defined as $s = 256\ x_i - x_i$. Division by 256 is performed by shifting the binary numbers eight positions to the right. The addition of k and s after being shifted results in a good approximation to expression (4) is

$$x_{i+1} = x_i + \frac{k - x_i}{256}. \tag{5}$$

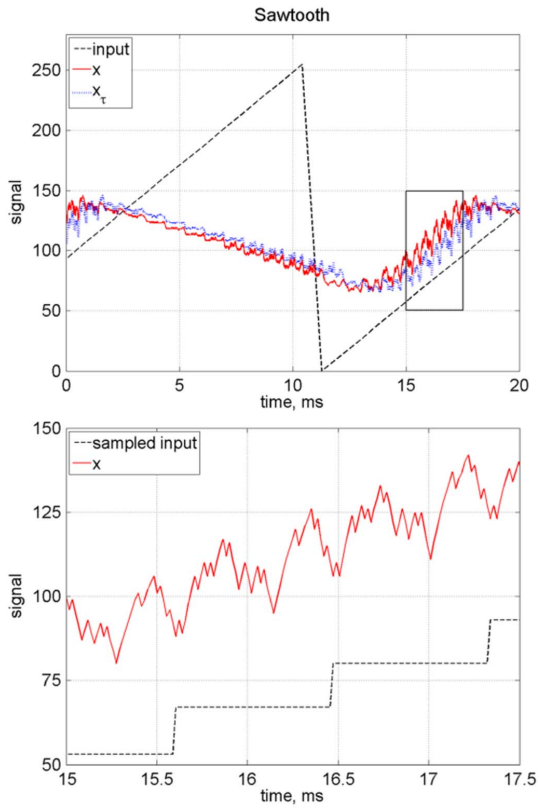

Fig. 4. Sawtooth input signal and its corresponding states $x(t)$ for the case in which $p = 7, \gamma = \eta = 0.9, \tau = 10$ T, $N = 50, \theta = 0.2$ T, and $T = 86.7\ \mu$s. The dynamical system nonlinearly maps each value of the sampled input into an N-dimensional state that facilitates classification. The bottom panel is an enlargement of $x(t)$, as indicated by the closed black box above, and the input expanded over a τ interval.

As an example of the implementation, we show in Fig. 4 a time trace of $x(t)$ when the input signal to the system $I(t)$ is a sawtooth wave. The bottom panel of Fig. 4 illustrates how each sample of the input signal is expanded over the N virtual nodes, multiplied by a random mask, and transformed by the Mackey–Glass nonlinearity.

III. Results

We synthesized the proposed digital circuitry on a Cyclone III low-cost FPGA. As shown in Table I, the implementation

TABLE I
HARDWARE RESOURCES (CYCLONE III EP3C16F484C6N)

Total logic elements (LE):	1,605/15,408 (10%)
Total combinational functions:	1,400/15,408 (9%)
Dedicated logic registers:	882/15,408 (6%)
Total memory bits:	768 + 16 c + 2816 τ/T
Embedded multiplier 9-bit element	16/112 (14%)

The parameters for the memory bits calculation are: c number of output classes and τ delay time.

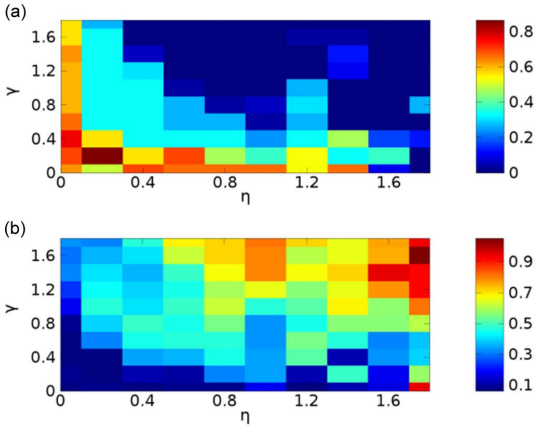

Fig. 5. Simulation results for the (a) error rate and (b) margin in the signal classification task. The two scanned parameters are γ and η . The fixed parameters are $p = 7$, $\tau = 10$ T, $N = 50$, and $c = 3$. The obtained error rate and margin are encoded in color.

needs limited hardware resources. The number of logic elements needed for the implementation represents 10% of the total number of logic elements in the FPGA, and only sufficient memory is needed to allocate the delayed signals. The implementation was designed to use delay values up to $\tau = 128T$ and a number of categories up to $c = 255$ using 71% of the RAM capacity. This limited use of resources highlights the advantages, from the hardware implementation point of view, of using a single nonlinear node with delay and time multiplexing. The measured power consumption of the whole FPGA development board (including peripherals) is 1.2 W, of which 83 mW is estimated to be due to the configured circuit.

As a proof-of-concept and to illustrate its real-time classification capabilities, we devised a simple benchmark task. We trained the reservoir to differentiate between three different noisy input signals, namely, sawtooth, sine, and square input waveforms.

In order to select the optimum configuration parameters, which would yield a good experimental classification performance, we carried out numerical simulations first. The performance of the system was evaluated in terms of the average error rate in the classification of the three input waveforms and the confidence margin, defined as the distance between the reservoir's best guess of the target and the closest competitor. In the Mackey–Glass nonlinearity, the parameters γ and η were varied, while the delay $\tau = 10$ T $(N = 50)$ and $p = 7$ were kept fixed. The robustness in the classification was tested by adding noise, with 2% relative amplitude, to the input signals and restricting the input (and output) values to a resolution of 8 bits.

Fig. 5 depicts the numerical results in the η–γ plane. It can be observed that large values of γ and η yield low error rates

TABLE II
PARAMETER VALUES USED IN THE EXAMPLE APPLICATION AND THEIR POSSIBLE RANGES FOR THE CURRENT FPGA IMPLEMENTATION

$\gamma = 0.9$	[0.0, 0.996]
$\eta = 0.9$	[0.0, 0.996]
$p = 7$	[2, 20] only integers
$\tau = 10T$	[0T, 128T]
T = 86.7 µs	$T = 255\, T_0\, T_{CLK}$
$T_0 = 17$	[10, 255]
N = 50	[1, 1023]
c = 3	[1, 255]
Input	[0.0, 0.996]

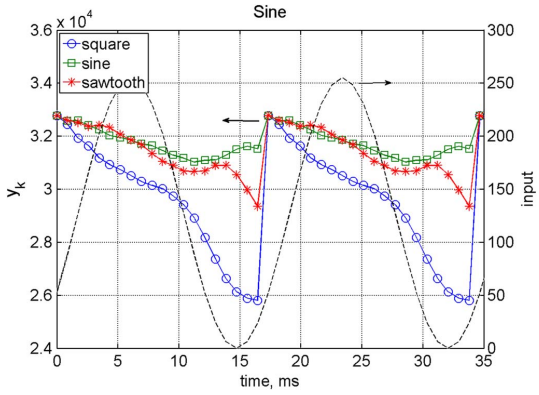

Fig. 6. Pattern recognition system behavior when the input is a sinusoidal signal. We show the input signal (right y-axis) along with the output classifiers (left y-axis) for the three possible patterns to be recognized. We obtain a clear recognition of the input type.

and a high separation between the three classes, allowing for good classification results since these conditions are the most robust to fluctuations. The optimum values for the configuration parameters compatible with our digital realization are shown in Table II. The parameters must be in the allowed ranges and ensure that the state x is kept within the limits given by the 8-bit resolution [0, 1).

The experimental realization yields the results depicted in Fig. 6. The output classifiers y_k, which are computed for each input step and updated at the end of each cycle (after 20 intervals τ), are plotted for the sinusoidal input case illustrating that the input signal can be clearly differentiated. Similar results are found for the sawtooth and rectangular patterns. Moreover, it is worth highlighting that this signal discrimination task is successfully performed in real time.

The system is trained such that the weighted sum of the states approximates the desired target value as closely as possible, following the standard procedure for RC [13], [14]. As for the numerical simulations, the input signal contained 2% random variations during the training process. The testing was carried out using previously unseen input data of the same kind and with the same noise level as those used in the training process. An error-free classification was found for a test set containing 1000 cycles of each type of the input signals.

The second task that we evaluated is a standard time-series prediction task. It consists in the one-step ahead prediction of the Santa Fe data set [16], an experimental recording of the output power of a far-infrared laser when operating in a chaotic regime. The data set contains 10 000 samples, where

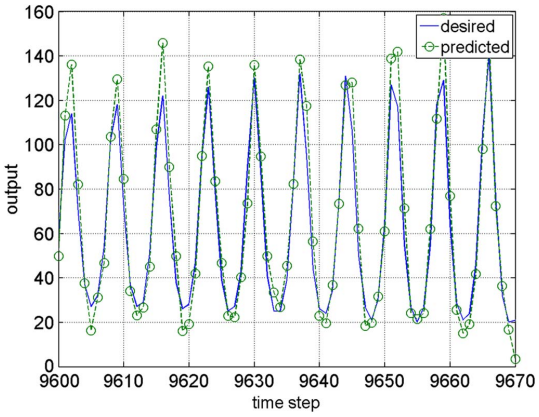

Fig. 7. Segment of the laser time series (predicted and targeted values).

the first 9000 were used for training and the remaining 1000 were used for testing. Fig. 7 shows a fragment of the predicted and targeted laser intensity values.

Similar simulations to the ones performed for the classification task were carried out to determine the configuration that gives the best performance. An optimum normalized mean square error of the prediction NMSE $= 0.051$ was obtained for $\gamma = 0.75$ and $\eta = 0.86$ (keeping the values of Table II for the rest of the parameters). This value increased to NMSE $= 0.107$ when the resolution of the state x values was limited to 8 bit. A comparable prediction error NMSE $= 0.131$ was obtained with our FPGA implementation.

Concerning the processing speed of the system, the characteristic time T is presented in Table II assuming a fixed integration step $\Delta t = T/255$ and a clock period $T_{\rm CLK} = 0.02$ μs. An additional T_0 factor was introduced in the system to ensure that the external memory had enough time to provide the mask values to the input layer block within one time step. This factor was also used to match the system's delay interval τ with the rate at which the input values were received via the serial port.

IV. Conclusion

The digital implementation of a single-node reservoir computer described here represents a novel platform to perform real-time pattern classification. As a result of using delay embedding and time multiplexing, the final solution can be developed within a simple low-cost FPGA. After optimizing the resources of the FPGA, the number of required logic elements is minimal. Therefore, this contribution extends the range of efficient implementations of ANNs in hardware [10], [17].

As an example, we have shown that this computationally low-cost method is capable of classifying different patterns, highlighting the capabilities of our simple approach. Importantly, this delay-based approach can be easily extended to differentiate a larger number of input classes. The properties of the system presented in this brief are particularly suited to process temporal information in, for example, distributed sensory networks, predictive controllers, and medical monitoring applications. In the latter case, a delay-based software implementation of RC was found to achieve state-of-the-art performance in the classification of electrocardiographic signals of cardiac arrhythmia with an average specificity of 97.75% and an average accuracy of 98.43% [18]. Since this medical application requires a sampling time of about 1 ms, ECG classification is fully compatible with our FPGA-based implementation of RC in real time.

References

[1] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.

[2] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.*, vol. 20, no. 3, pp. 391–403, Apr. 2007.

[3] M. Lukoševičius, H. Jaeger, and B. Schrauwen, "Reservoir computing trends," *KI—Künstliche Intelligenz*, vol. 26, no. 4, pp. 365–371, Jan. 2012.

[4] L. Appeltant *et al.*, "Information processing using a single dynamical node as complex system," *Nat. Commun.*, vol. 2, pp. 468–472, 2011.

[5] L. Larger *et al.*, "Photonic information processing beyond Turing: An optoelectronic implementation of reservoir computing," *Opt. Exp.*, vol. 20, no. 3, pp. 3241–3249, Jan. 2012.

[6] Y. Paquot *et al.*, "Optoelectronic reservoir computing," *Sci. Rep.*, vol. 2, p. 287, 2012.

[7] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nat. Commun.*, vol. 4, p. 1364, 2013.

[8] M. C. Soriano *et al.*, "Delay-based reservoir computing: Noise effects in a combined analog and digital implementation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 2, pp. 388–393, Feb. 2015.

[9] R. V. Kulkarni, A. Forster, and G. K. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A survey," *IEEE Commun. Survey Tuts.*, vol. 13, no. 1, pp. 68–96, 1st Quart. 2011.

[10] J. Mishra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1–3, pp. 239–255, Dec. 2010.

[11] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. Van Campenhout, "Compact hardware liquid state machines on FPGA for real-time speech recognition," *Neural Netw.*, vol. 21, no. 2/3, pp. 511–523, Mar./Apr. 2008.

[12] N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, "Reservoir computing with a single time-delay autonomous Boolean node," *Phys. Rev. E, Statist, Nonliear, Soft Matter Phys.*, vol. 91, no. 2, 2015, Art. ID. 020801.

[13] H. Jaeger, "The 'echo state' approach to analyzing and training recurrent neural networks," German Nat. Res. Center Inf. Technol., St. Augustin, Germany, Tech. Rep. GMD Rep. 148, 2001.

[14] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science* vol. 304, no. 5667, pp. 78–80, 2004.

[15] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, pp. 287–289, Jul. 1977.

[16] A. S. Weigend and N. A. Gershenfeld, Time Series Prediction: Forecasting the Future and Understanding the Past, 1993. [Online]. Available: http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html

[17] S. Decherchi, P. Gastaldo, A. Leoncini, and R. Zunino, "Efficient digital implementation of extreme learning machines for classification" *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 8, pp. 496–500, Aug. 2012.

[18] M. Escalona-Moran, M. C. Soriano, I. Fischer, and C. R. Mirasso, "Electrocardiogram classification using reservoir computing with logistic regression," *IEEE J. Biomed. Health Informat.*, vol. 19, no. 3, pp. 892–898, May 2015.