technologies

MDPI

*Article*

# FPGA-Based Implementation of a Multilayer Perceptron Suitable for Chaotic Time Series Prediction

**Ana Dalia Pano-Azucena [1], Esteban Tlelo-Cuautle [1,*], Sheldon X. -D. Tan [2], Brisbane Ovilla-Martinez [3] and Luis Gerardo de la Fraga [4]**

1 Department of Electronics, INAOE, Puebla 72840, Mexico; ana.dalia.p.a@gmail.com
2 Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521, USA; stan@ece.ucr.edu
3 Unidad Iztapalapa, Universidad Autónoma Metropolitana, Mexico City 09340, Mexico; bris.ovilla@gmail.com
4 Department of Computer Sciences, CINVESTAV, Mexico City 07360, Mexico; fraga@cs.cinvestav.mx
* Correspondence: etlelo@inaoep.mx; Tel.: +52-222-2663100

Received: 16 August 2018; Accepted: 28 September 2018; Published: 1 October 2018


**Abstract:** Many biological systems and natural phenomena exhibit chaotic behaviors that are saved in time series data. This article uses time series that are generated by chaotic oscillators with different values of the maximum Lyapunov exponent (MLE) to predict their future behavior. Three prediction techniques are compared, namely: artificial neural networks (ANNs), the adaptive neuro-fuzzy inference system (ANFIS) and least-squares support vector machines (SVM). The experimental results show that ANNs provide the lowest root mean squared error. That way, we introduce a multilayer perceptron that is implemented using a field-programmable gate array (FPGA) to predict experimental chaotic time series.

**Keywords:** chaos; time series prediction; FPGA; multilayer perceptron

## 1. Introduction

Chaotic time series prediction techniques have the challenge of forecasting long horizons of observed data, which must guarantee low error that can be measured as root mean squared error (RMSE). Among the kinds of chaotic signals that require attention to predict their future behavior are for example those associated to human diseases and pathologies, namely: epilepsy, pancreatic beta cell, electroencephalogram, electrocardiogram, and so on. Besides, predicting chaotic time series is a challenge because they are unpredictable data. Nowadays, many researchers are investigating the appropriate prediction technique for this kind of time series. In the recent work [1], the authors showed the prediction capabilities of three techniques, namely: artificial neural networks (ANNs), the adaptive neuro-fuzzy inference system (ANFIS) and least-squares support vector machines (LS-SVM). From the experiments, SVM and ANFIS were overcome by ANNs. Those experimental results are used herein as the base to highlight that ANNs are a good option for chaotic time series prediction. There may exist other time series prediction techniques, all of them focused on predicting future events using past information. However, only a few techniques can be successfully applied in the long-term prediction of chaotic time series, due to problems such as: accumulation of errors, lack of information and sensitivity to initial conditions.

In the state of the art, one can find several techniques in order to solve the problem on chaotic time series prediction considering short-term and long-term prediction horizons, e.g., ANNs [2–4], fuzzy systems [5–7], SVM [8,9], hybrid systems [10–12], etc. ANNs have been employed in [2] to predict

the spontaneous fluctuation in vascular caliber; in that work, the authors basically used a multilayer perceptron (MLP). In [13], the authors showed that the dynamics of a chaotic system are present in the time series analysis of the river flow, and by combining phase-space reconstruction and ANN techniques, they predicted the daily river flow. In [14], it was shown that financial time series have chaotic behavior, and they can be predicted using an MLP. On the other hand, SVM has been widely used in regression and classification problems, and its principles can easily be extended to time series prediction, as shown in [8]. The authors in [15] predicted the global incident solar radiation applying SVM, and also in [16], it was applied to model river water pollution. The other prediction technique applied herein is ANFIS, which has emerged as a synergic hybrid intelligent system because it combines the human-like reasoning style of a fuzzy logic system with the learning and computational capabilities of ANNs [17]. In [18] the authors reported the prediction of chaotic time series of Mackey and Glass, the Dow Jones Company and the Mexican exchange stock.

This article considers as cases of study different chaotic time series generated by a chaotic oscillator and with different values of their maximum Lyapunov exponent (MLE). In chaos theory, it is well known that the higher the MLE value, the higher the unpredictability of a chaotic time series. We show preliminary results already provided in [1] by applying three techniques: ANNs, LS-SVM and ANFIS. As ANNs showed the lowest RMSE for predicting the same chaotic time series, then we introduce an MLP implemented in a field-programmable gate array (FPGA). This electronic system can find application in healthcare informatics [19].

Section 2 describes the generation of chaotic time series from a chaotic oscillator implemented in an FPGA, and their analysis to evaluate the MLE is performed by using the free and online available time series analysis (TISEAN) tool. Section 3 describes three time series prediction techniques: ANNs, SVM and ANFIS. Their comparison is given in Section 3.4. Section 4 introduces an MLP implemented in an FPGA and shows experimental prediction results. Finally, the conclusions are listed in Section 5.

## 2. Generating Chaotic Time Series

The time series data that are used herein to predict their future behavior are generated by the chaotic oscillator described by:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -ax - by - cz + d_1 f(x) \end{aligned} \tag{1}$$

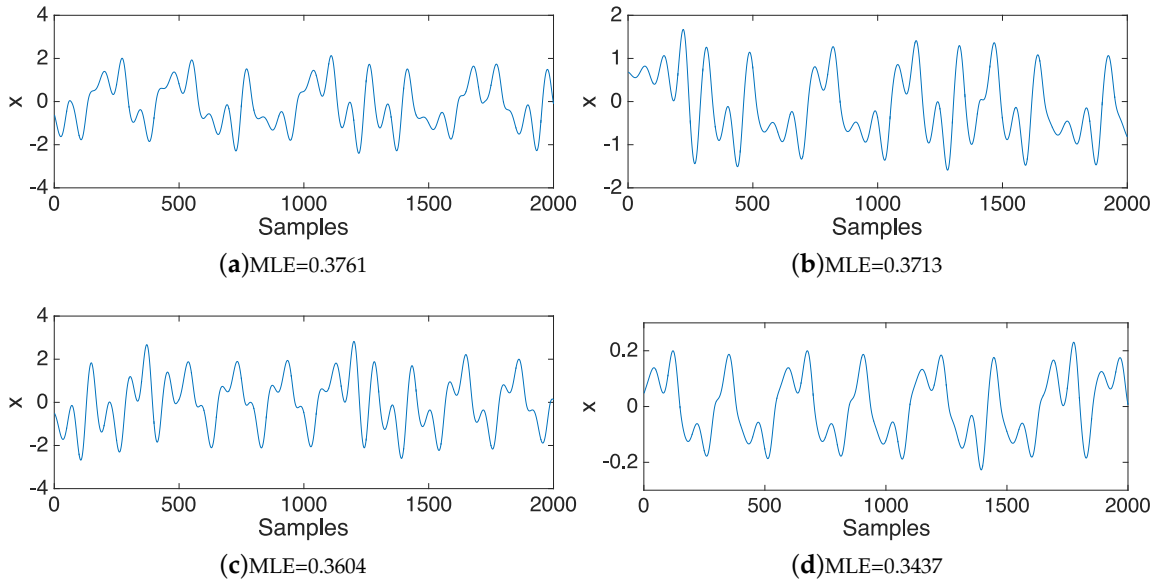
where $x$, $y$, $z$ are the state variables; $a$, $b$, $c$, $d_1$ are real and positive coefficients; and $f(x)$ is a nonlinear function described by (2) with $s(x) = 60.606$ as the slope, $\alpha = 0.0165$ the break points and $k = 1$ the saturation level.

$$f(x) = \begin{cases} k & x < \alpha \\ s(x) & -\alpha \leq x \leq \alpha \\ -k & x < -\alpha \end{cases} \tag{2}$$

This chaotic oscillator was implemented using an FPGA, as already shown in [20]; their coefficients $a$, $b$, $c$, $d_1$ were varied as shown in [21–23] in order to maximize MLE, which is related to the unpredictability of the chaotic time series data. For instance, Table 1 lists the coefficient values associated with eight different MLE values. Figure 1 shows examples of chaotic time series corresponding to the state variable $x$ and with different values of MLE. In this work, the time series data of this state variable $x$ are used to predict future behavior of six and 12 steps ahead. The amplitudes of all the chaotic time series data are normalized in the range $[\pm 1]$.

**Table 1.** Optimized maximum Lyapunov exponent (MLE) for a two-scroll attractor and their associated ($a$, $b$, $c$, $d_1$) coefficient values.

| Case | $a$ | $b$ | $c$ | $d_1$ | MLE |
|---|---|---|---|---|---|
| 1 | 1.0000 | 1.0000 | 0.4997 | 1.0000 | 0.3761 |
| 2 | 1.0000 | 0.7884 | 0.6435 | 0.6665 | 0.3713 |
| 3 | 0.8661 | 1.0000 | 0.3934 | 0.9903 | 0.3607 |
| 4 | 0.7746 | 0.6588 | 0.5846 | 0.4931 | 0.3460 |
| 5 | 1.0000 | 0.7000 | 0.6780 | 0.1069 | 0.3437 |
| 6 | 1.0000 | 0.7000 | 0.7000 | 0.2542 | 0.3425 |
| 7 | 0.5610 | 0.9470 | 0.3460 | 0.6810 | 0.2225 |
| 8 | 0.7000 | 0.7000 | 0.7000 | 0.7000 | 0.1117 |




(**a**)MLE=0.3761 (**b**)MLE=0.3713

(**c**)MLE=0.3604 (**d**)MLE=0.3437

**Figure 1.** Examples of the chaotic time series with different values of MLE.

## 3. Chaotic Time Series Prediction Techniques

Time series prediction techniques can be as simple as predicting one future value, known as single step ahead prediction, or complex so as to predict more than one future value, which can be described by $\hat{y}(t+k)$ to save a specific input vector at any time $t$. That way, we are interested in predicting horizons of six $k = 6$ and twelve $k = 12$ steps ahead. In this manner, the format of training data is described by (3) and (4), respectively. In both cases, the prediction of six and 12 steps ahead is verified by evaluating the root mean squared error (RMSE) described by (5), which has been used to measure the prediction performance of ANNs, ANFIS and SVM techniques in [1].

The chaotic time series data are divided into two sets: training and validation sets. In those cases, we used 2000 samples for training the prediction technique (ANN, ANFIS and SVM) and 2000 samples for validation. In the training phase, the parameters of the prediction techniques are adjusted. In the validation phase, the prediction errors are analyzed when new data are used.

$$\hat{y}(t+6) = x_1(t), x(t-6), x(t-12), x(t-18) \quad (3)$$

$$\hat{y}(t+12) = x_1(t), x(t-12), x(t-24), x(t-36) \quad (4)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2} \quad (5)$$

### *3.1. Artificial Neural Networks*

An ANN is a set of elementary processing units called neurons whose processing capability is stored in the connections by synaptic weights and whose adaptation depends on learning. Figure 2 shows the structure of an artificial neuron where $x_{1,2,..,j}$ represents the input, which has an associated weight $w$. The artificial neuron has a bias ($\gamma$), and $f$ denotes the activation function. That way, the state of an artificial neuron is evaluated by (6).

$$f(u) = \sum_{i=1}^{n} x_j w_j + b \tag{6}$$

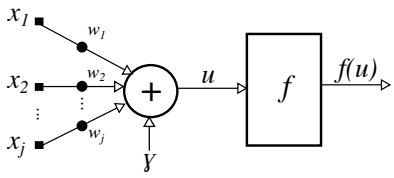



**Figure 2.** Structure of an artificial neuron.

A particular case of an ANN is the multilayer perceptron (MLP), which consists of an input and an output layer, and one or more hidden layers. In an MLP, the input signals propagate through the network layer-by-layer [24]. In an MLP, the numbers of neurons and layers can be the difference between success and failure [25]. For instance, one guideline to choose the number of hidden neurons is the application of the geometric pyramid rule given in [25], where the number of neurons follows a pyramid shape. First, one must set the number of neurons in the input layer $n$ and the number of neurons in the output layer $m$. Table 2 lists several equations associated with different numbers of hidden layers ($h_i$). As one can see, for a three-layer network with $n$ input neurons and $m$ output neurons, the hidden layer has $\sqrt{n \cdot m}$ neurons. A similar rule applies to networks with $i$ hidden layers; however, the evaluation of the number of hidden layer neurons is slightly more complex.

**Table 2.** Computation of the number of hidden layer neurons using the geometric pyramid rule.

| Hidden Layers | Geometric Pyramid Rule | |
|---|---|---|
| 1 | $r = (n \cdot m)^{1/2}$ | $h_1 = r$ |
| 2 | $r = (n/m)^{1/3}$ | $h_1 = m \cdot r^2$, $h_2 = m \cdot r$ |
| 3 | $r = (n/m)^{1/4}$ | $h_1 = m \cdot r^3$, $h_2 = m \cdot r^2$, $h_3 = m \cdot r$ |
| ⋮ | ⋮ | ⋮ |
| $i$ | $r = (n/m)^{1/i+1}$ | $h_1 = m \cdot r^i$, $h_2 = m \cdot r^{i-1}$, ... , $h_i = m \cdot r$ |

In this work, we designed two MLPs from the application of the geometric pyramid rule. The first one consists of five layers, with sixteen neurons in the input layer ($n = 16$), three hidden layers ($h_1$, $h_2$, $h_3$) and one neuron in the output layer $m = 1$. Therefore, from Table 2: $r = 2$ and $h_1 = 8$, $h_2 = 4$, $h_3 = 2$. The second MLP consists of six layers with 32 neurons in the input layer ($n = 32$), four hidden layers and one neuron in the output layer $m = 1$, so that $r = 2$ and $h_1 = 16$, $h_2 = 8$, $h_3 = 4$, $h_4 = 2$.

A third MLP introduced in [26] is also used herein. It consists of six layers, with different numbers of neurons. Table 3 lists the characteristics of the three MLPs that are used herein to predict chaotic time series, where the neurons of the input and hidden layers have a hyperbolic tangent function ($h$) and the neuron of the output layer has a linear function ($l$).

**Table 3.** Characteristics of three MLPs: the one from [26] and the others designed by applying the geometric pyramid rule with five and six layers.

| Characteristics | MLP from [26] | MLP Applying the Geometric Pyramid Rule | |
|---|---|---|---|
| Number of layers | 6 | 5 | 6 |
| Number of neurons | $4 \times 7 \times 4 \times 8 \times 5 \times 1$ | $16 \times 8 \times 4 \times 2 \times 1$ | $32 \times 16 \times 8 \times 4 \times 2 \times 1$ |
| Activation function | $h \times h \times h \times h \times h \times h \times l$ | $h \times h \times h \times h \times l$ | $h \times h \times h \times h \times h \times l$ |
| Learning algorithm | Levenberg–Marquardt algorithm | | |

### 3.2. Adaptive Neuro-Fuzzy Inference System

ANFIS is a type of system that incorporates ANNs and fuzzy logic. In this case, the neurons of the ANN incorporate TSK (Takagi–Sugeno–Kang) so that the system is adaptive, fault tolerant, distribute, learns from experience and presents generalizability. Assuming that the fuzzy control system has an output $f$ and two inputs $(x; y)$, thus the two fuzzy rules are represented by (7).

$$\begin{aligned} R^1 &: \text{IF } x \text{ is } A_1 \text{ AND } y \text{ is } B_1, \text{ THEN } f_1 = p_1 x + q_1 y + r_1 \\ R^2 &: \text{IF } x \text{ is } A_2 \text{ AND } y \text{ is } B_2, \text{ THEN } f_2 = p_2 x + q_2 y + r_2 \end{aligned} \tag{7}$$

Figure 3 shows the general structure of ANFIS, where the function of nodes in each layer is:

- Layer 1 contains input nodes passing external signals to the next layer.
- Layer 2 performs the parameter adjustment of the input membership function.
- Layers 3 and 4 perform fuzzy operations.
- Layers 5 and 6 weight and provide the output of the system.

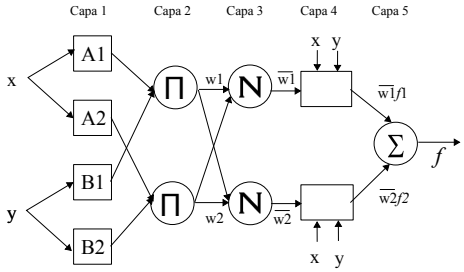



**Figure 3.** General structure of ANFIS.

In ANFIS, four inputs and $2^4$ = 16 rules are used. Furthermore, genfis1is used to generate an initial membership function matrix (generalized bell function) from the training data.

### 3.3. Support Vector Machine

The prediction of time series using least-squares support vector machines (LS-SVM) is done using a set of historical values of time series as inputs and a single output as the target value. When the prediction of the time series is developed by SVMs, given time series data $y_t$ composed by a set of consecutive real values $y_t = \{y_t \in \mathbb{R} \mid y_1, y_2, ..., y_n\}$, the prediction is represented by dividing $y_t = \{y_1, y_2, ..., y_n\}$ in windows $w = (y_t, ..., y_{t+p-1})$ of size $p$, in order to find a function $f(y_t, ..., y_{t+p-1}) = y_{t+p}$. The radial basis function is used as a kernel function because it provides excellent performance under assumptions of general softness [27]. In the space of characteristics, LS-SVM has the form $f(\boldsymbol{x}) = \boldsymbol{w}^T \varphi(\boldsymbol{x}) + b$, where $\boldsymbol{w}$ is a vector of weights. Figure 4 shows a functional structure detailed in [28].

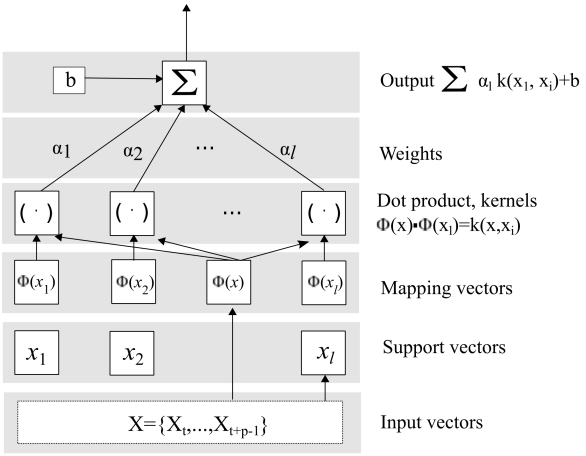



**Figure 4.** SVM functional structure [28].

### *3.4. Chaotic Time Series Prediction by ANN, ANFIS and SVM*

The three MLPs from Table 3 are compared to ANFIS and SVM for the prediction of the eight chaotic time series data listed in Table 1. Table 4 lists the prediction error of the eight chaotic time series associated with different MLE values. The RMSE was obtained by using a prediction horizon of six steps ahead. Figure 5 shows the performance comparison of the prediction techniques with respect to their RMSE value obtained during the validation stage. As one sees, the three MLPs produce lower error. Comparing the MLPs listed in Table 3, it is observed that the six-layer and five-layer topologies, both designed from the geometric pyramid rule, provide the least prediction error.

**Table 4.** RMSE obtained during the validation stage with a six steps ahead prediction horizon.

| MLE | ANN | | | ANFIS | LS-SVM |
|---|---|---|---|---|---|
| | 5 Layers | 6 Layers | [26] | | |
| 0.1117 | $7.31 \times 10^{-4}$ | $3.91 \times 10^{-4}$ | $1.70 \times 10^{-3}$ | 0.0153 | 0.0120 |
| 0.2225 | $3.00 \times 10^{-3}$ | $1.80 \times 10^{-3}$ | $5.49 \times 10^{-3}$ | 0.0460 | 0.0180 |
| 0.3425 | $1.52 \times 10^{-3}$ | $2.98 \times 10^{-4}$ | $1.83 \times 10^{-3}$ | 0.0276 | 0.0012 |
| 0.3437 | $8.78 \times 10^{-4}$ | $3.59 \times 10^{-4}$ | $1.80 \times 10^{-3}$ | 0.0244 | 0.0041 |
| 0.3460 | $1.09 \times 10^{-3}$ | $5.14 \times 10^{-4}$ | $2.09 \times 10^{-3}$ | 0.0180 | 0.0103 |
| 0.3607 | $3.22 \times 10^{-3}$ | $2.29 \times 10^{-3}$ | $5.03 \times 10^{-3}$ | 0.0446 | 0.0816 |
| 0.3713 | $1.03 \times 10^{-3}$ | $9.63 \times 10^{-4}$ | $2.17 \times 10^{-3}$ | 0.0315 | 0.0037 |
| 0.3761 | $1.93 \times 10^{-3}$ | $8.14 \times 10^{-4}$ | $2.87 \times 10^{-3}$ | 0.0373 | 0.0062 |

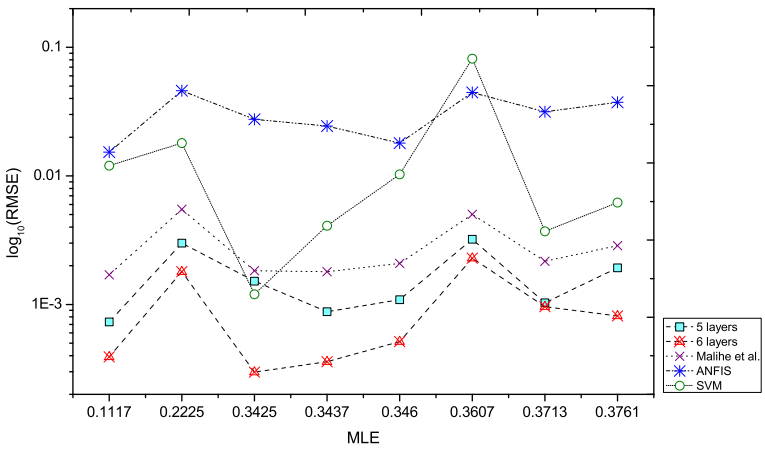



**Figure 5.** Comparing the RMSE of the MLPs from Table 3, ANFIS and SVM for the prediction of the chaotic time series listed in Table 1 and achieving a six steps ahead prediction horizon [1].

Considering the previous results, the MLPs with five layers and six layers are selected to predict the eight chaotic time series, but now to achieve a twelve steps ahead horizon. Figure 6 shows the RMSE obtained during the validation stage, again for the eight MLE values associated with the chaotic time series from Table 1.

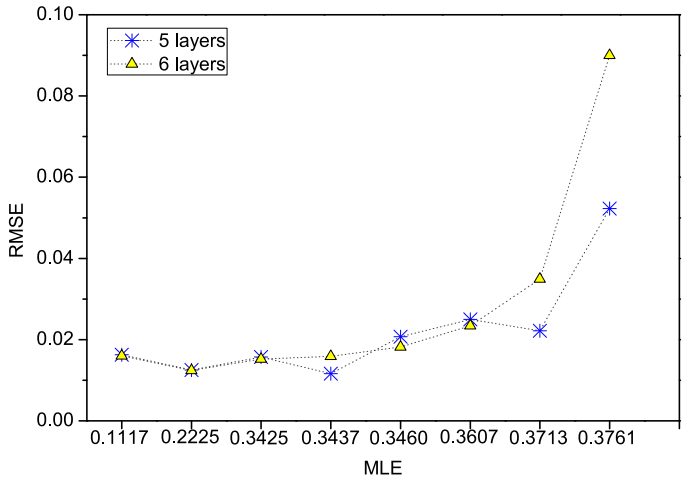



**Figure 6.** Comparing the RMSE of the MLPs with five and six layers from Table 3 for the prediction of the chaotic time series listed in Table 1 and achieving a twelve steps-ahead prediction horizon.

## 4. FPGA Implementation of an MLP

This section shows the FPGA implementation of the MLP with five layers shown in Figure 7, which follows the methodology already given in [20]. In this MLP, the information flows in one direction, so that the prediction depends largely on the input signals. The MLP is interfaced with a personal computer using the serial communication protocol to feed the input data, as described in [20]. The MLP is implemented using the FPGA Ciclone IV GX FPGA DE2i-150 from Altera. The descriptions of the digital blocks are performed using fixed-point notation, which provides a high speed and a low cost of hardware resources. In this manner, $N = 32$ bits are used to represent a fixed point format, where the most significant bit represents the sign, three bits are used to represent the integer part and 28 bits are used to represent the fractional part. This allows representing numbers over the range $\{-2^{N-1}, 2^{N-1} - 1\}$, with a numerical resolution equal to $3.7253 \times 10^{-9}$, enough to represent the weight and bias of the neurons in the MLP.

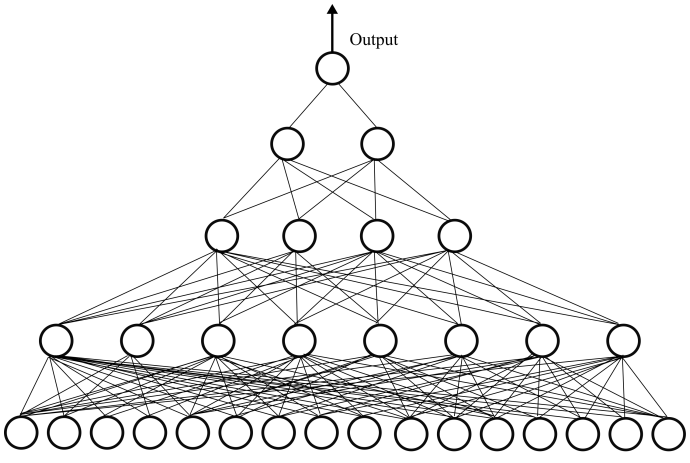



**Figure 7.** Details of the MLP with five layers.

From Figure 2, one can identify the basic building blocks of a neuron, namely: multiplication, sum and a block dedicated to implementing the hyperbolic tangent function. The VHDL block descriptions of the adder and multiplier are shown in Figure 8; they perform operations with 2'scomplement, and they have a clock (CLK) and a reset (RST) pin to make the system synchronous.

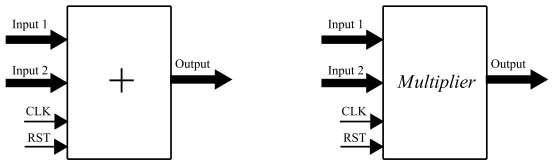



**Figure 8.** VHDL block descriptions of the adder and multiplier including clock and reset pins. CLK, clock; RST, reset.

The hyperbolic tangent activation function can be approached by using the piecewise-linear (PWL) approximation technique developed by [29] by using segments of order two, as shown in (8), where $\beta$ and $\theta$ determine the slope and gain $H_{s1}(z)$ between $-L \leq z \leq L$. In this manner, the hyperbolic tangent is approached herein by (9), where $\theta = 0.25$, $L = 2$ and $\beta = 1$.

$$H_{s1}(w) = \begin{cases} w(\beta - \theta w) & \text{for } 0 \leq w \leq L \\ w(\beta + \theta w) & -L \leq w \leq 0 \end{cases} \tag{8}$$

$$G_{s1}(w) = \begin{cases} 1 & \text{for } L \leq w \\ H_{s1} & \text{for } -L < w < L \\ -1 & w \leq -L \end{cases} \tag{9}$$

Figure 9 shows the block description to implement the hyperbolic tangent activation function using PWL functions described by (9). The value of $H_{s1}(w)$ is obtained by simplifying operations; therefore, it can be expressed as $w(\beta \pm \theta w)$, where an extra multiplier allows one to perform the conditions described by (8). That is, when the product $\theta w$ is positive, then $H_{s1}(w) = w(\beta - \theta w)$, and when the product $\theta w$ is negative, then $H_{s1}(w) = w(\beta + \theta w)$. On the other hand, the value of $G_{s1}(w)$ is obtained using a comparator block.

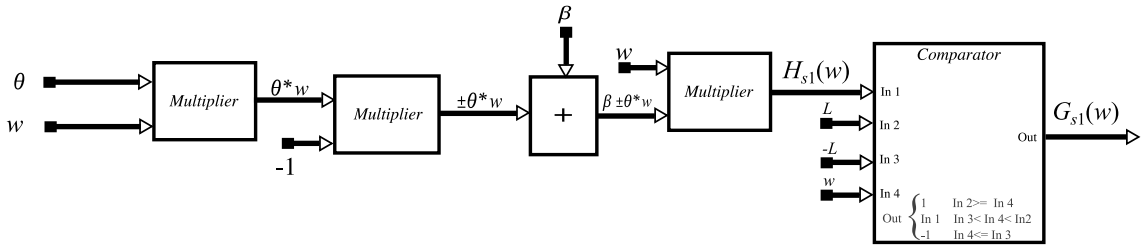



**Figure 9.** Block description of the hyperbolic tangent activation function using piecewise-linear (PWL) functions.

Figure 10 shows the synthesis of an artificial neuron. Recall that the design of each neuron depends on the number of inputs, weights, biases and activation function. That way, the MLP shown in Figure 7 has 30 neurons between the first and fourth layers, which are synthesized by using the neuron from Figure 10, because all 30 neurons have an activation function of a hyperbolic type, as listed in Table 3. The last layer has an activation function of a linear type, so that the neuron is modified at $f(u)$ from Figure 10.

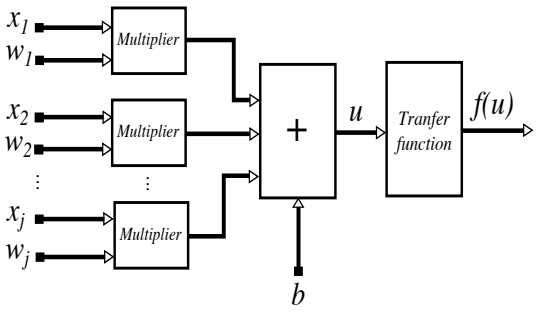



**Figure 10.** Synthesis of the functional structure of an artificial neuron.

Figure 11 shows the experimental results of the five-layer MLP, using the validation dataset and six steps ahead prediction with the chaotic time series with MLE = 0.3761. The MLP achieves an RMSE = 0.019.

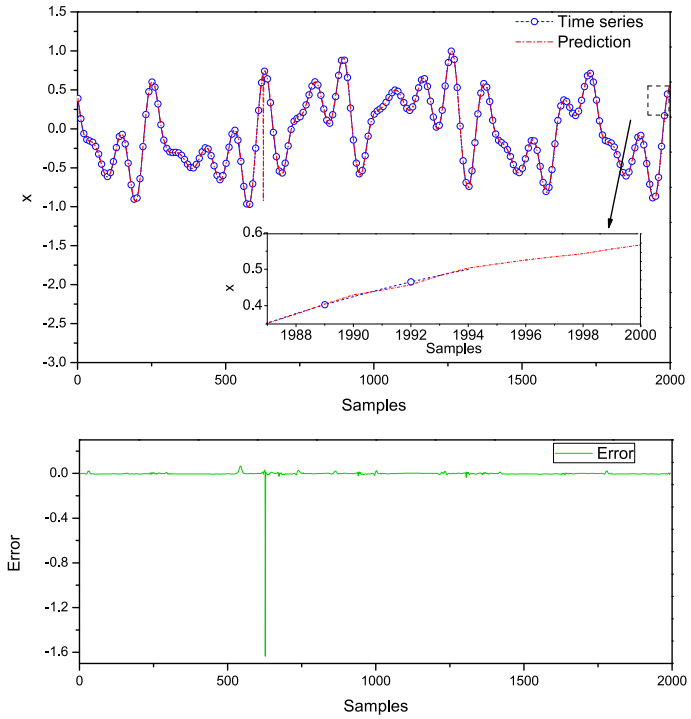



**Figure 11.** Six steps ahead prediction result of the five-layer MLP for the chaotic time series with MLE = 0.3761.

Figure 12 shows the experimental results of the five-layer MLP, using the validation dataset and twelve steps ahead prediction with the chaotic time series with MLE = 0.3761. The MLP achieves an RMSE = 0.0266.

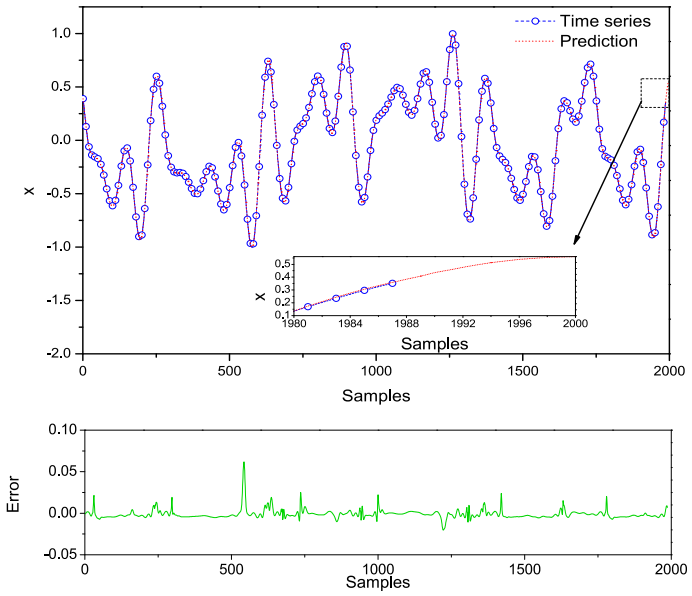



**Figure 12.** Twelve steps ahead prediction result of the five-layer MLP for the chaotic time series with MLE = 0.3761.

## 5. Conclusions

This article showed the performance comparison among three MLPs, LS-SVM and ANFIS to predict chaotic time series that were generated from a chaotic oscillator implemented in an FPGA. Eight chaotic time series were generated with different MLE values, and the prediction techniques were compared with respect to the RMSE. As MLPs showed the least RMSE, we introduced the FPGA implementation of an MLP with five layers designed by applying the geometric pyramid rule. The experimental results were performed for a prediction horizon of six and twelve steps ahead. The MLP predicted the chaotic time series with a very low RMSE. This confirmed the usefulness of the FPGA to implement an MLP to predict chaotic time series data.

**Author Contributions:** Investigation, A.D.P.-A., E.T.-C., S.X.-D.T., B.O.-M. and L.G.d.l.F. Writing, review and editing, E.T.-C. These authors contributed equally to this work.

**Funding:** This research was funded by CONACyT at Mexico under Grant Number 237991 and by UC-MEXUSunder Grant CN-16-161.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Pano-Azucena, A.D.; Tlelo-Cuautle, E.; Tan, S.X.D. Prediction of chaotic time series by using ANNs, ANFIS and SVMs. In Proceedings of the 7th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 7–9 May 2018; pp. 1–4.
2. Stamatis, N.; Parthimos, D.; Griffith, T.M. Forecasting chaotic cardiovascular time series with an adaptive slope multilayer perceptron neural network. *IEEE Trans. Biomed. Eng.* **1999**, *46*, 1441–1453. [CrossRef] [PubMed]
3. Ishikawa, M.; Moriyama, T. Prediction of time series by a structural learning of neural networks. *Fuzzy Sets Syst.* **1996**, *82*. [CrossRef]
4. Aras, S.; Kocakoç, İ.D. A new model selection strategy in time series forecasting with artificial neural networks: IHTS. *Neurocomputing* **2016**, *174*, 974–987. [CrossRef]
5. Pouzols, F.M.; Lendasse, A.; Barros, A.B. Autoregressive time series prediction by means of fuzzy inference systems using nonparametric residual variance estimation. *Fuzzy Sets Syst.* **2010**, *161*, 471–497. [CrossRef]

6. Di Martino, F.; Loia, V.; Sessa, S. Fuzzy transforms method in prediction data analysis. *Fuzzy Sets Syst.* **2011**, *180*, 146–163. [CrossRef]
7. Singh, P.; Borah, B. High-order fuzzy-neuro expert system for time series forecasting. *Knowledge-Based Syst.* **2013**, *46*, 12–21. [CrossRef]
8. Thissen, U.; Van Brakel, R.; De Weijer, A.; Melssen, W.; Buydens, L. Using support vector machines for time series prediction. *Chemom. Intell. Lab. Syst.* **2003**, *69*, 35–49. [CrossRef]
9. Kim, K.j. Financial time series forecasting using support vector machines. *Neurocomputing* **2003**, *55*, 307–319. [CrossRef]
10. Ardalani-Farsa, M.; Zolfaghari, S. Chaotic time series prediction with residual analysis method using hybrid Elman–NARX neural networks. *Neurocomputing* **2010**, *73*, 2540–2553. [CrossRef]
11. Bodyanskiy, Y.; Vynokurova, O. Hybrid adaptive wavelet-neuro-fuzzy system for chaotic time series identification. *Inf. Sci.* **2013**, *220*, 170–179. [CrossRef]
12. Bagheri, A.; Peyhani, H.M.; Akbari, M. Financial forecasting using ANFIS networks with quantum-behaved particle swarm optimization. *Expert Syst. Appl.* **2014**, *41*, 6235–6250. [CrossRef]
13. Delafrouz, H.; Ghaheri, A.; Ghorbani, M.A. A novel hybrid neural network based on phase space reconstruction technique for daily river flow prediction. *Soft Comput.* **2018**, *22*, 2205–2215. [CrossRef]
14. Ravi, V.; Pradeepkumar, D.; Deb, K. Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms. *Swarm Evol. Comput.* **2017**, *36*, 136–149. [CrossRef]
15. Deo, R.C.; Wen, X.; Qi, F. A wavelet-coupled support vector machine model for forecasting global incident solar radiation using limited meteorological dataset. *Appl. Energy* **2016**, *168*, 568–593. [CrossRef]
16. Kisi, O.; Parmar, K.S. Application of least square support vector machine and multivariate adaptive regression spline models in long term prediction of river water pollution. *J. Hydrol.* **2016**, *534*, 104–112. [CrossRef]
17. Al-Mahasneh, M.; Aljarrah, M.; Rababah, T.; Alu'datt, M. Application of Hybrid Neural Fuzzy System (ANFIS) in Food Processing and Technology. *Food Eng. Rev.* **2016**, *8*, 351–366. [CrossRef]
18. Melin, P.; Soto, J.; Castillo, O.; Soria, J. A new approach for time series prediction using ensembles of ANFIS models. *Expert Syst. Appl.* **2012**, *39*, 3494–3506. doi:10.1016/j.eswa.2011.09.040. [CrossRef]
19. Pano-Azucena, A.D.; Tlelo-Cuautle, E.; Tan, S. Electronic System for Chaotic Time Series Prediction Associated to Human Disease. In Proceedings of the 2018 IEEE International Conference on Healthcare Informatics (ICHI), New York, NY, USA, 4–7 June 2018; pp. 323–327.
20. Tlelo-Cuautle, E.; de la Fraga, L.; Rangel-Magdaleno, J. *Engineering Applications of FPGAs*; Springer: New York, NY, USA, 2016.
21. de la Fraga, L.G.; Tlelo-Cuautle, E.; Carbajal-Gómez, V.; Munoz-Pacheco, J. On maximizing positive Lyapunov exponents in a chaotic oscillator with heuristics. *Rev. Mex. Fis.* **2012**, *58*, 274–281.
22. Carbajal-Gómez, V.H.; Tlelo-Cuautle, E.; Fernández, F.V. Optimizing the positive Lyapunov exponent in multi-scroll chaotic oscillators with differential evolution algorithm. *Appl. Math. Comput.* **2013**, *219*, 8163–8168. [CrossRef]
23. Carbajal-Gómez, V.H.; Tlelo-Cuautle, E.; Fernández, F.V. Application of computational intelligence techniques to maximize unpredictability in multiscroll chaotic oscillators. In *Computational Intelligence in Analog and Mixed-Signal (AMS) and Radio-Frequency (RF) Circuit Design*; Springer: New York, NY, USA, 2015; pp. 59–81.
24. Shiblee, M.; Kalra, P.K.; Chandra, B. Time Series Prediction with Multilayer Perceptron (MLP): A New Generalized Error Based Approach. In *Advances in Neuro-Information Processing*; Köppen, M., Kasabov, N., Coghill, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 37–44.
25. Masters, T. *Practical Neural Network Recipes in C++*; Morgan Kaufmann: Massachusetts, MA, USA, 1993.
26. Molaie, M.; Falahian, R.; Gharibzadeh, S.; Jafari, S.; Sprott, J.C. Artificial neural networks: powerful tools for modeling chaotic behavior in the nervous system. *Front. Comput. Neurosci.* **2014**, *8*, 40. [CrossRef] [PubMed]
27. Ye, Y.M.; Wang, X.D. Chaotic time series prediction using least squares support vector machines. *Chin. Phys.* **2004**, *13*, 454.

28. Sapankevych, N.I.; Sankar, R. Time Series Prediction Using Support Vector Machines: A Survey. *IEEE Comput. Intell. Mag.* **2009**, *4*, 24–38. [CrossRef]
29. Kwan, H. Simple sigmoid-like activation function suitable for digital hardware implementation. *Electron. Lett.* **1992**, *28*, 1379–1380. [CrossRef]