Chapter

Real-Time Echo State Network Based on FPGA and Its Applications

Yongbo Liao

Abstract

In this chapter, a hardware processing architecture of real-time echo state network based on field-programmable gate array (FPGA) is proposed, which solves the problem that it is difficult to obtain the output weight of the network in real time. The design of this architecture strictly follows the reservoir calculation (RC) theory, and its five components are established in FPGA: input module, reservoir module, output module, training module, and system switch module. This paper implements the architecture in Altera FPGA chip and verifies it through the application of pattern recognition, waveform generation, and multiple-input multiple-output (MIMO) channel prediction. Experimental results show that the hardware-implemented real-time echo state network can identify the duty cycle of different input signals, generate floating-point waveforms, and predict the MIMO channel by training. In this paper, a real-time echo state network based on field programmable gate array is proposed, which has the advantages of fast computation speed, less resource consumption, and ideal simple task execution.

Keywords: FPGA, ESN, pattern recognition, waveform generation, MIMO channel prediction, real-time, training, testing

1. Introduction

Echo state network [1] simplifies training tasks into linear regression tasks. It mainly solves the problems of large consumption of Recurrent Neural Network (RNN) training resources, long running time, and slow convergence. There are many studies on the applications of echo state network, such as wind power ramp time prediction [2, 3], medical image recognition classification [4], water flow prediction [5], etc. There are also many studies on the structure of the echo state network, such as the dynamic reservoirs that increase their stochastic properties [6] or delay characteristics [7], correlation entropy replaces traditional error function [8], change calculation model [9, 10], etc. Less research work on hardware platform implementation of neural networks, such as [11] proposed a software framework for simulating RNN circuits, [12, 13] proposed FPGA/software framework; however these frameworks are always trained in software such as MATLAB, which not be strictly said to be hardware implementation. The FPGA-based real-time echo state network structure proposed in this chapter trains the output weights on the FPGA platform without calculating the relevant parameters by means of software. In

order to verify the performance of the proposed architecture, two types of benchmark tasks were performed: the output signal which was a binary signal [14] and a floating point number signal and a MIMO channel prediction task [15].

2. Theory and model

This section describes the mathematical model of the echo state network. The structural model is shown in **Figure 1**. Let the model have K input units whose vector form is

$$u(n) = (u_1(n), u_2(n) \cdots\cdots\cdots u_K(n))^T \tag{1}$$

N reservoir units, the vector form is

$$x(n) = (x_1, x_2 \cdots\cdots\cdots x_N)^T \tag{2}$$

L output units whose vector form is

$$y(n) = (y_1, y_2 \cdots\cdots\cdots y_L)^T \tag{3}$$

where $(\bullet)T$ is the transpose, n is the discrete time, and the input/reservoir/output connection weight is represented by a weight matrix of size N × K/N × N/L×(K + N), i.e.

$$W^{in} = \left(w_{i,j}^{i,n}\right), W = (w_{i,j}), W^{out} = \left(w_{ij}^{out}\right) \tag{4}$$

The output unit can select whether to feed back to the intermediate unit and the connection weight is represented by a feedback weight matrix of size N × L:

$$W^{\text{back}} = \left(w_{ij}^{back}\right) \tag{5}$$

Intermediate cell status updates according to the formula

$$x(n+1) = f\left(W^{in}u(n+1) + Wx(n) + W^{back}y_{\text{target}}(n)\right) \tag{6}$$

where u(n + 1) is the external given input at time n + 1, such as Eq. (1); $y_{\text{target}}(n)$ is the ideal output at time n, in the form of Eq. (2); and f represents the transfer

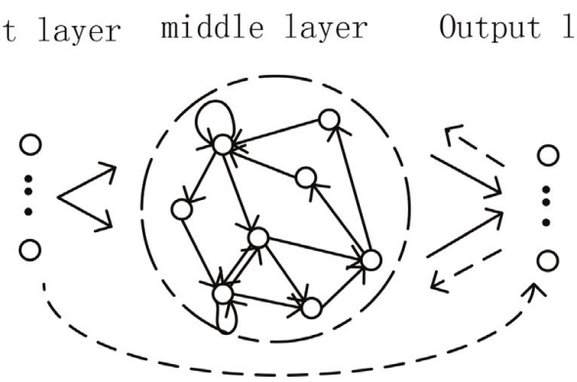

Figure 1.
The basic structure of the echo state network. The arrows indicate the flow of data.

function of the intermediate unit, mainly using the S function, but sometimes a linear network $f = 1$ is also used. Output calculation is based on

$$y(n+1) = f_{out}(W^{out}(u(n+1), x(n+1))) \tag{7}$$

where $(u(n+1), x(n+1))$ represents the juxtaposition of the input and the intermediate state vector, as shown in the input layer to the output layer of the dashed arrow in **Figure 1**, in some applications, such as [16], where the data stream does not exist. That is, the output is calculated directly using the intermediate state value. The output transfer function is usually $f_{\text{out}} = \tanh$ or $f_{\text{out}} = 1$, depending on whether the output unit is nonlinear or linear. The output weight W^{out} is calculated according to the following formula:

$$W^{out} = Y_{target} X^T \left(XX^T + \alpha^2 I\right)^{-1} \tag{8}$$

where $I \in R^{N \times N}$ is the identity matrix, α is the regularization factor, $R \in R^{N \times N}$ is the set matrix of $(u(n+1), (x(n+1)))$, Y_{target} is the ideal output set matrix, and $(\bullet)^{-1}$ is the matrix inversion.

3. Real-time FPGA echo state network structure

Real-time FPGA echo state network execution structure maps Eqs. (6)–(8) to six modules, which are input module, reservoir module, output module, training module, system switch module, and random number generator, as shown in **Figure 2**. The input module is a two-input single-output module, and the input is a random input weight Win generated by a random number generator and an external signal $u(n + 1)$, performing a W in $u(n + 1)$ multiplication operation, and encoding the input signal to form a data signal that can be calculated by the reservoir module. The reservoir module is a five-input single-output module that strictly performs the remainder of Eq. (6), the input including the encoded external signal from the input module, reservoir initial state value (this input is the state value of the previous clock reservoir module output as the operation progresses), network expected output, reservoir interconnection weights generated by the random number generator, and feedback. Connecting the weight, output high-dimensional state signal, the specific circuit is shown in **Figure 3**.

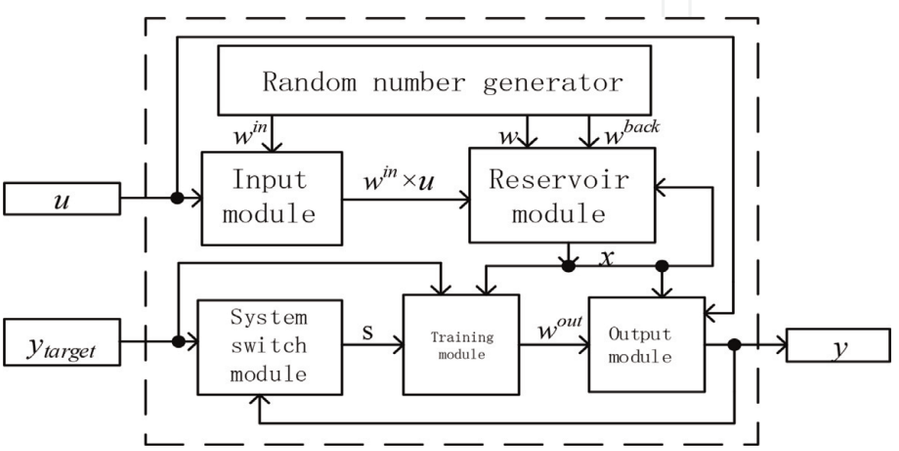

Figure 2.
Real-time FPGA echo state network structure diagram.

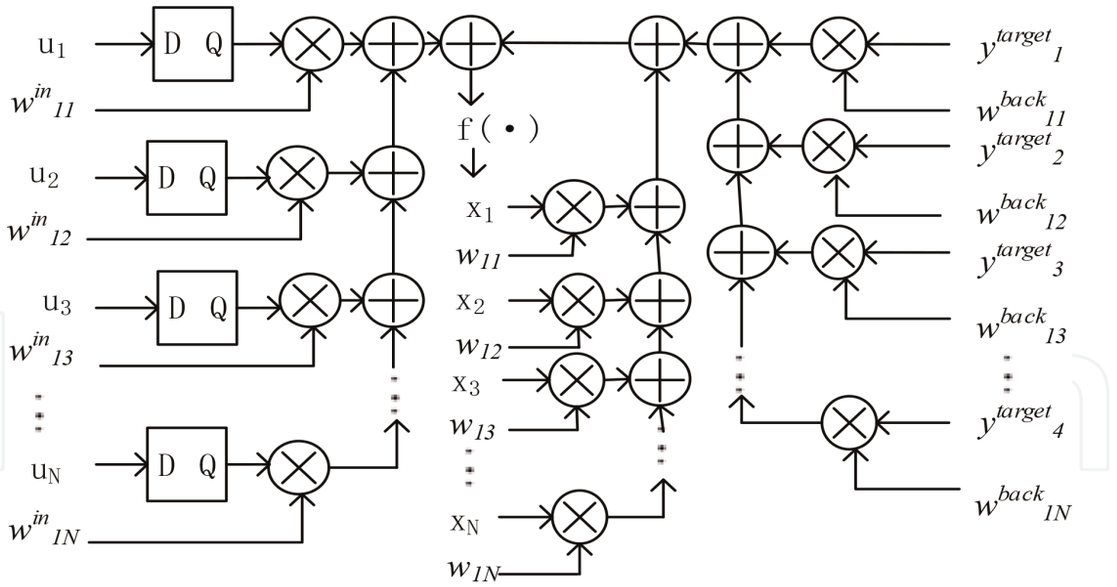

Figure 3.
Reservoir module.

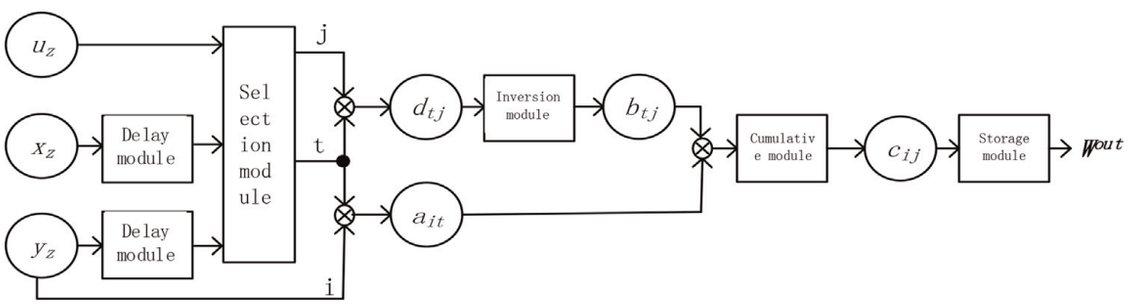

Figure 4.
Training module. Subscript z represents the zth unit of input, reservoir, and output.

(**Figure 3** circuit is the circuit that outputs the reservoir state x1, other state circuits are similar). The output module is a three-input single-output module, the input is an external input signal, the state value is obtained by the reservoir module calculation, and the output connection weight is calculated by the training module. The output is the final acquired network output. The function is to perform simple multiplication and addition on the input data and decode the result to form an output signal. The training module is a three-input single-output module, the input is an externally given desired output, a status signal is generated by the reservoir module, and an enable signal S is sent by the system switch module, When $S = 0$, the module stops working. When $S = 1$, the module works, and the output is the core parameter output connection weight of the echo state network. The function is expressed as Eq. (8). The specific implementation mechanism is shown in **Figure 4** (in Eq. (8)), $Y_{target}X^T = (a_{it})$, $XX^T + \alpha^2 I = (d_{tj})$, $(XX^T + \alpha^2 I)^{-1} = (b_{tj})$, $W^{out} = (c_{ij})$; then, the output weight $W^{out} = (c_{ij})$ is obtained; The system switch module is a two-input single-output module. The input is a network output signal and a desired output signal. The output is an enable signal for controlling the operation of the training module. Its function is mainly to judge the network performance. When the network output signal matches the expected output signal or the difference is within the receiving range, the output $S = 0$ and the other cases continue to output $S = 1$. A random number generator is used to generate inputs, reservoirs, and feedback weights.

The following sections detail how to train a real-time FPGA echo state network. As is well known, FPGAs implement digital systems that primarily process digital

signals, providing a logic cell array that can be configured as a given function via a bitstream file. Its basic digital logic, the smallest programmable logic unit, is the logic gate. Therefore, FPGAs are the best device for performing echo state networks. The architecture is all implemented in the FPGA. On the one hand, the dynamic reservoir (i.e., the middle layer) is established. On the other hand, how to obtain the real-time output weights when the input signal is the digital signal "0" or "1" is established.

The dataflow and training process of the real-time FPGA echo state network structure will be briefly described as follows:

Given: Input and target output sequence $u(n)$ and $y_{\text{target}}(n)$, n $= 1 \cdots\cdots T$.

Objective: The teacher signal input/output training acquires W^{out} and acquires the network output signal by loading the input signal.

Proceed as follows:

The random number generator module generates an input, a reservoir, and a feedback weight of the echo state network and sends the input and feedback weights to the input module, and the reservoir weight is sent to the reservoir module.

The input module loads the input signal, encodes the input signal, and sends it to the reservoir module.

The reservoir module receives the encoded signal sent by the input module, loads the target output signal, acquires the feature value, and sends it to the output module and the training module.

The training module loads the input signal, the target output signal, and the characteristic value sent by the reservoir module; calculates the output weight; sends the result to the output module; and determines whether to stop training according to the signal sent by the system switch module.

The output module loads the input signal and calculates the network output according to the characteristic value sent by the reservoir module and the output weight sent by the training module and sends the network output value to the system switch module and outputs the actual network value.

The system switch module loads the target output signal and the network output signal to determine whether the match is matched and sends the judgment result back to the training module. If it matches, it sends back $S = 0$; if it does not match, it sends back $S = 1$;

Repeat steps 2–6 until the judgment result of the system switch module is to stop training, that is, $S = 0$.

This is followed by the regular echo state network function, which only performs input, reservoir, and output modules and outputs network prediction values.

4. Experiment and analysis

A total of two types of benchmark task experiments were performed: the output signal was a binary signal and a floating point number signal and a multiple-input multiple-output channel prediction task experiment. The programming language is Verilog HDL, the chip uses Altera Stratix III FPGA, and the integrated place and route are implemented in QUARTUS II.

4.1 Different duty cycle signal pattern recognition

The pattern recognition reference task with different duty cycles is very similar to the memory resistance based on reservoir calculation (RC) in [14]. The mode signals with different duty cycles are shown in **Figure 5**. The input signal of the echo

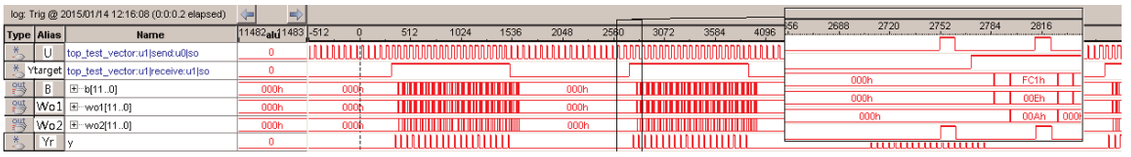

Figure 5.
Different duty cycle pattern recognition training.

state network is *U*, the expected output is *Y* target, and the actual output signal is Yr; Wo1 and Wo2 are output weights, and B is the bias signal. The second line (signal Y_target) represents the expected response of the first line (signal U). When the duty cycle of the input signal is less than 50%, the signal Y_target should converge to 0 and should converge to 1 for a duty cycle greater than 50%. As shown in **Figure 5**, the online training echo state network in the FPGA obtains Wo1, Wo2, and B, and their values are 00Eh, 00Ah, and FC1h, respectively. After the output weight is obtained, different duty cycle mode signals are loaded into the trained echo state network, and the result is as shown in **Figure 6**. The output signal (Yr) changes between a duty cycle of the input signal (U) greater than 50% and less than 50%.

Figure 7 is a graph showing the percentage of the total number of nerve cells implemented in the FPGA logic and FPGA and the error curve. It can be seen that the logic utilization is less than 60% until the number of neurons is 512 units. When the number of nerves exceeds 16 units, the error between the actual output signal and the ideal output is zero. Therefore, the circuit resources proposed in this paper consume less, and the convergence speed is faster.

4.2 Sine wave generator

Here we test how to train the echo state network to generate a sine wave signal, which is a floating point number, which is very similar to the simple sine wave test

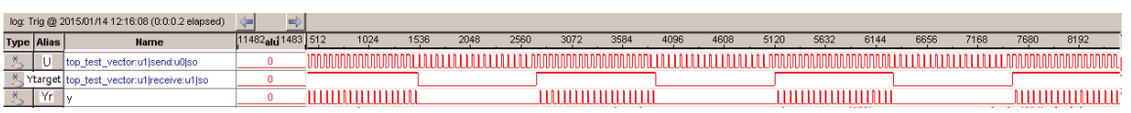

Figure 6.
Echo state network test results.

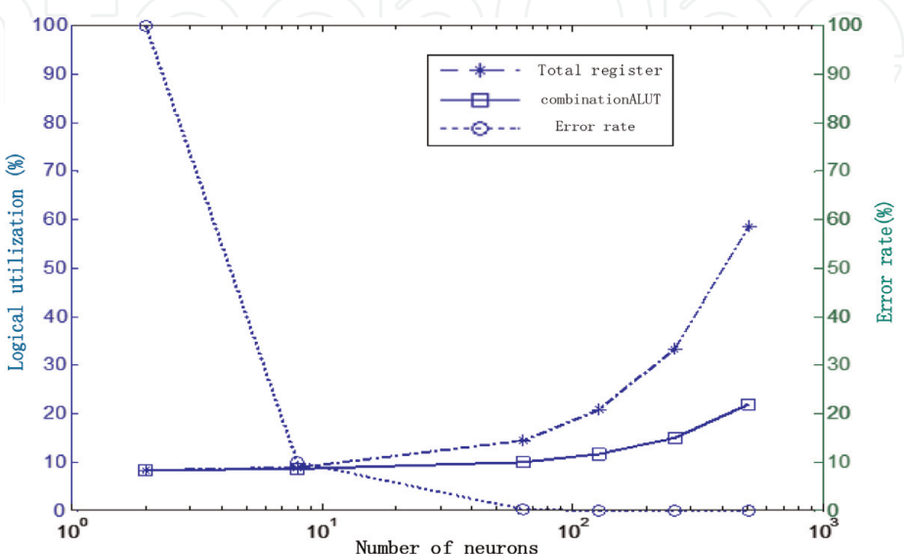

Figure 7.
Relationship between neuron number and logic utilization and error curve.

performed in MATLAB when the echo state network is proposed in [1]. The ideal sine wave signal is given by the equation $y_{target}(t) = \sin(t/8)$. There is no input in this experiment, so set W^{in} to 0; W and W^{back} are matrices formed by random numbers, and the basic unit of the echo state network is the standard S unit (i.e., the activation function is S function tanh), and the training result is shown in **Figure 8**. In the process of generating sine wave training, the ideal output signal (Y_{target}) is a floating point number, so the actual output signal (Yr), the normalized root mean square error signal (E), and the output weight signal (Wout1, Wout2, Wout3, Wout4) are all floating point numbers. The training error is calculated in the system switch module, and E = 3D0228E7h is calculated according to the normalized root mean square error calculation equation $E = \sqrt{\sum_{n=1}^{N} \frac{\left(y_{target}(n) - y(n)\right)^2}{N \times \mathrm{var}\left(y_{target}\right)}}$. The optimized output weights Wout1, Wout2, Wout3, and Wout4 of the training module are BDFEDD9Dh, 3CB22AA8h, 3CA0BDD8h, and 3F55E542h, respectively. The waveform shown in **Figure 9** is acquired by the SignalTap II logic analyzer and is a floating-point sine wave generated for the trained echo state network. The Altera floating-point IP core was used in the experiment to set up the echo state network. As shown in **Figure 10**, the top is the Y target signal, the middle is the actual output y signal of the network, and the bottom is the error waveform of the network expected output and the actual output.

4.3 MIMO channel prediction

Recurrent neural networks have been widely used in MIMO systems [17–23]. Echo state networks are a way to train recurrent neural networks. They have faster convergence characteristics, and more efficient tracking channel state changes than other traditional training methods. For a 2 × 2 multiple-input multiple-output system with a binary phase shift keying (BPSK) modulator, as shown in **Figure 11**, the zero-forcing equalizer used in the receiver section can reduce symbol interference (ISI) due to the precise channel. It is estimated that the zero-forcing equalization can be improved by the degraded radio channel, and therefore the proposed architecture is used for MIMO channel prediction.

The matrix equation of the MIMO system shown in **Figure 11** is given as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \tag{9}$$

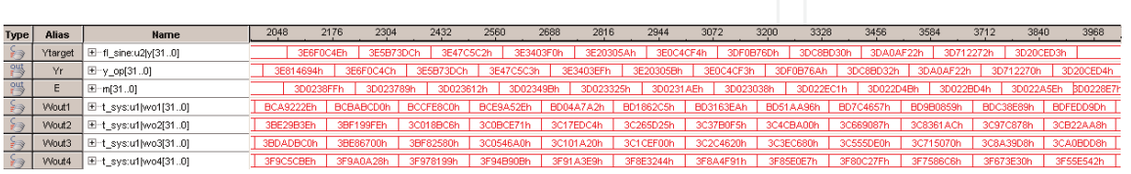

Figure 8.
Sine wave generation training.

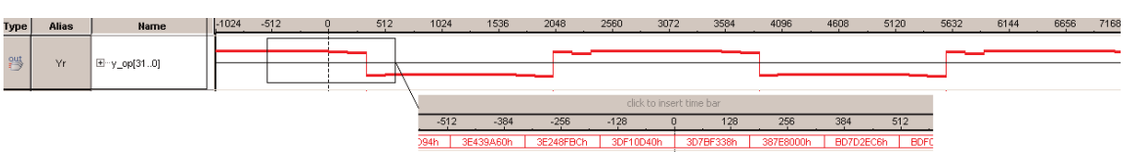

Figure 9.
Echo state network generating floating point sine wave.

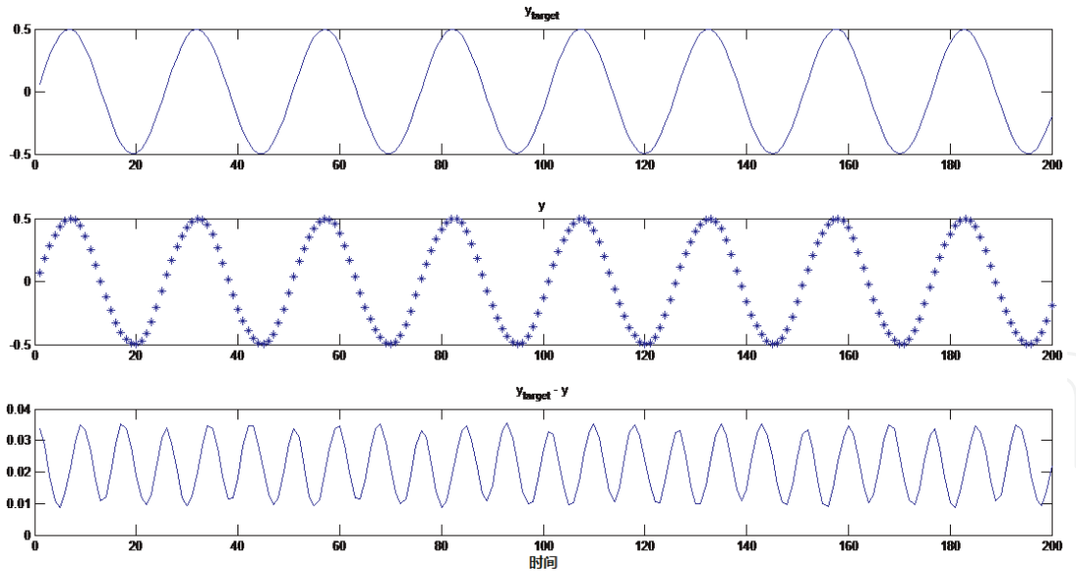

Figure 10.
Target signal (y_{target}), actual signal (y), and error curve (number of neurons is 8).

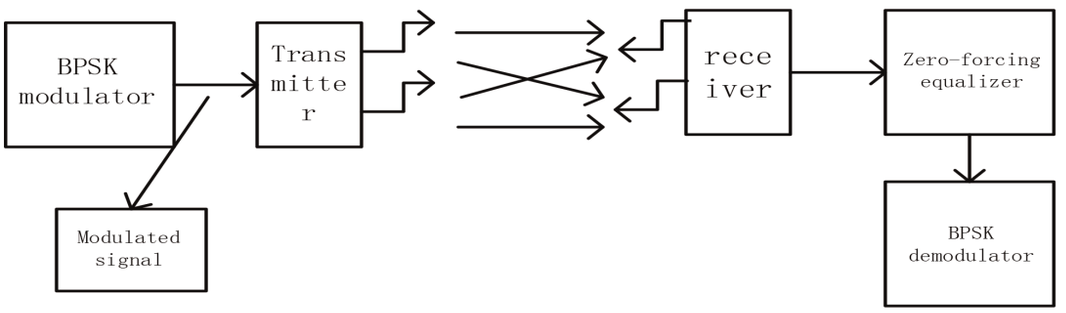

Figure 11.
2 × 2 MIMO system.

The system can be represented as a compact form $Y = HX + n$, where Y is a 2×1 received signal vector, H is a 2×2 channel coefficient matrix, X is a 2×1 propagation vector, and n is a 2×1 additive white Gaussian noise vector. The channel is considered to be a Rayleigh decay with a mean of 0 and a variance of 0.5. At the receiving end, the zero-forcing equalization performs the prediction of the propagated signal, and the equation is

$$\hat{X} = \left(H^H H\right)^{-1} HY \tag{10}$$

where $\left(H^H H\right)^{-1} H$ represents the pseudo inverse of H. The predicted $\hat{X}$ loaded BPSK demodulator recovers the original information.

In order to dynamically update the channel state at each step, an echo state network is added to the 2×2 MIMO system in **Figure 11**. The system structure diagram after adding the echo state network is shown in **Figure 12**. The echo state network channel prediction strategy is shown in **Figure 13**. The channel coefficients are trained in the echo state network channel prediction. Once the training is completed, the echo state network channel prediction can automatically generate the predicted channel coefficients, and the predicted channel coefficients are loaded into the zero-forcing equalization, thereby completing the MIMO channel prediction.

Figure 14 shows the RTL level circuit diagram generated by the FPGA. The system mainly includes a transmitter (fx), a receiver (rx), and an echo state

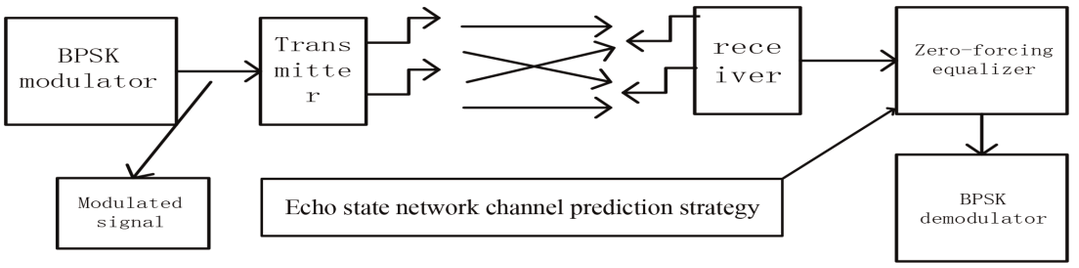

Figure 12.
Echo state network for 2 × 2 MIMO systems.

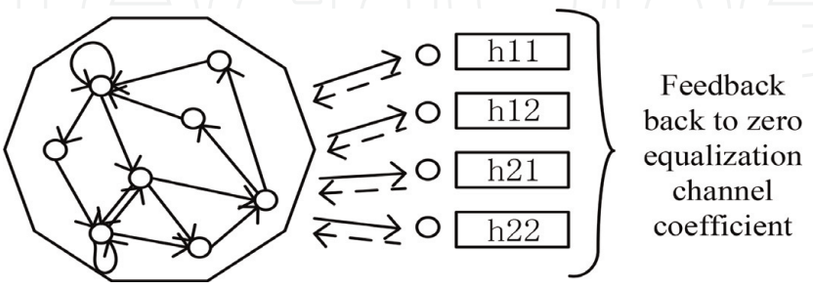

Figure 13.
Echo state network channel prediction strategy structure diagram.

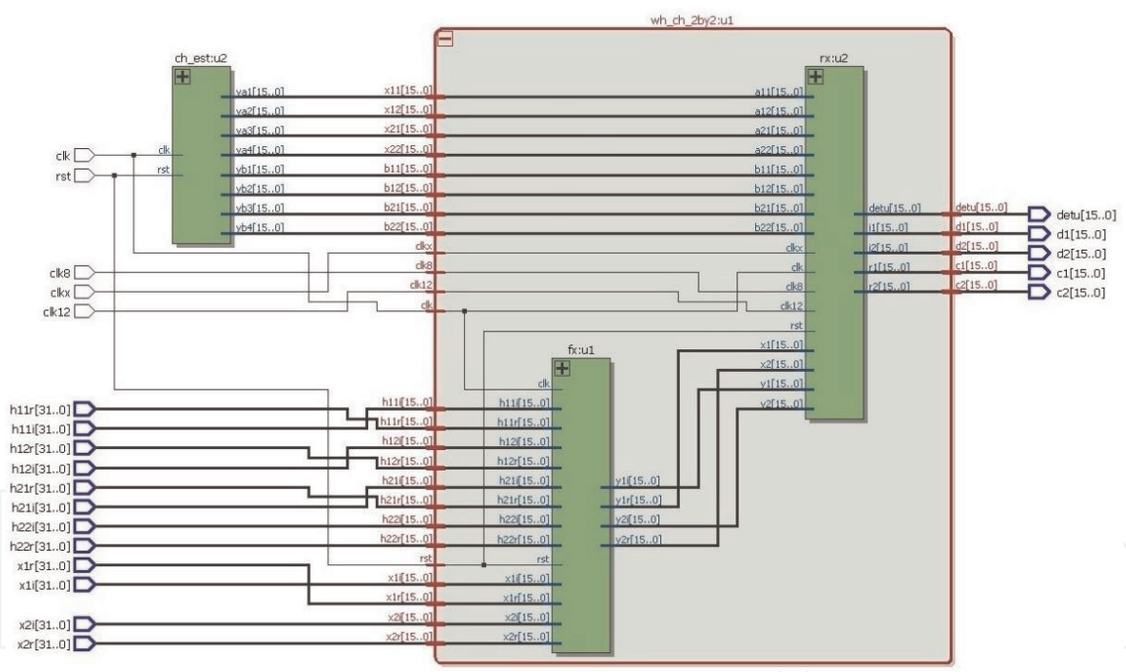

Figure 14.
RTL circuit diagram obtained by FPGA synthesis.

network module (ch_test). In the transmitting module (fx), the signals x1r, x2r and x1i, x2i are the real and imaginary parts of the MIMO system input signals X1 and X2, respectively, and h11r, h12r, h21r, h22r and h11i, h12i, h21i, h22i are, respectively, MIMO channels. The real and imaginary parts of the coefficient, y1r, y2r and y1i, y2i are the real and imaginary parts of the received signal, respectively. In the receiving module (rx), a11, a12, a21, a22 and b11, b12, b21, b22 are the real and imaginary parts of the channel prediction coefficients, and the output is calculated by the echo state network module (ch_test), c1, c2, and d1. d2 is the real and imaginary part of $\hat{X}$, respectively, corresponding to the demodulated signal X, and the signal "detu" is the demodulation scale factor of c1, c2, d1 and d2.

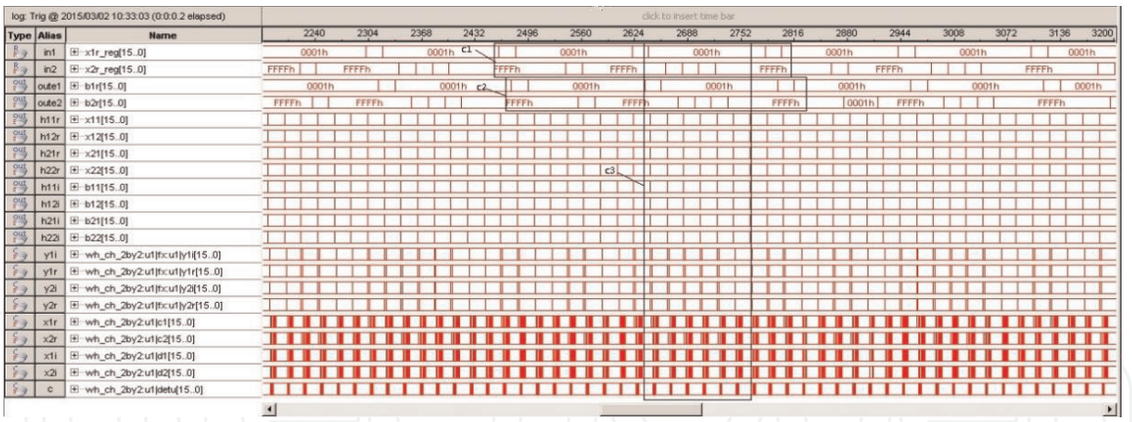

Figure 15.
Real-time waveforms for channel prediction in MIMO systems and echo state networks.

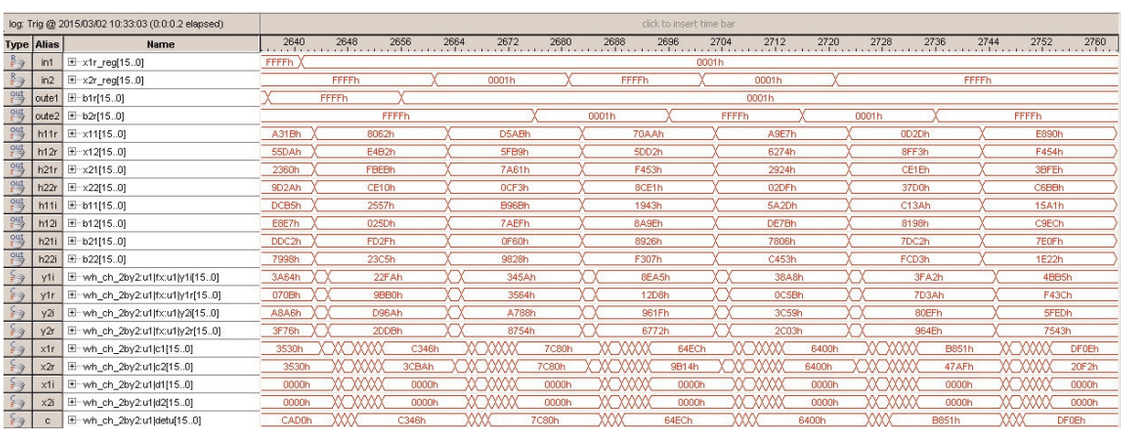

Figure 16.
C3 partial enlargement result.

The designed MIMO system is downloaded to the FPGA chip through a bitstream file, and the waveform result is obtained by a SignalTap II logic analyzer, as shown in **Figure 15**. It can be seen from the waveform diagram that the signal waveforms of the C1 and C2 parts are completely identical. In the C1 portion, the signals in1 and in2 correspond to x1r and x2r, respectively, and in the C2 portion, the signals oute1 and oute2 correspond to the real part of the signal X. X1r, x2r, and x1i, x2i are the real and imaginary parts before demodulation of the demodulated signal, respectively.

In order to be able to explain the C3 part, the C3 part is enlarged here (see **Figure 16**). As can be seen from the C3 section, the received signals Y1 and Y2 and the estimated channel matrix are all changed. However, zero-forcing equalization can still predict values x1r, x2r, and C through the echo state network. After the processing is completed, BPSK demodulation signals "oute1" and "oute2" are obtained.

5. Conclusion

The real-time FPGA echo state network structure is proposed and studied. The input weight and the reservoir weight are randomly determined before training, and the output weight is calculated in real time in the FPGA by training the echo state network. In the above two benchmark experiments (pattern recognition and waveform generation) and MIMO channel prediction experiments, the proposed hardware architecture can recognize the duty cycle of different input signals, generate floating point waveforms, and predict channel coefficients. From the

experimental results, the echo state network is faster, the resources are less occupied, and the simple task execution is ideal. In future research work, the proposed FPGA real-time echo state network will be used in more complex 5G-based wireless MIMO-OFDM systems.

Author details

Yongbo Liao
State Key Laboratory of Electronic Thin Films and Integrated Devices, University of Electronic Science and Technology of China, Chengdu, China

*Address all correspondence to: lyb@uestc.edu.com

References

[1] Jaeger H. The "Echo State" Approach to Analysing and Training Recurrent Neural Networks. German National Research Center for Information Technology; 2001

[2] Dorado-Moreno M, Cornejo-Bueno L, Gutiérrez PA, et al. Robust estimation of wind power ramp events with reservoir computing. Renewable Energy. 2017;**111**:428-437

[3] Escalona-Morán MA, Soriano MC, Fischer I, et al. Electrocardiogram classification using reservoir computing with logistic regression. IEEE Journal of Biomedical and Health Informatics. 2015;**19**(3):892-898

[4] Bezerra SGTA, Andrade CBD, Valença MJS. Using reservoir computing and trend information for short-term streamflow forecasting. In: Artificial Neural Networks and Machine Learning —ICANN 2016. Springer International Publishing; 2016

[5] Basterrech S, Rubino G. Echo state queuing networks: A combination of reservoir computing and random neural networks. Probability in the Engineering & Informational Sciences. 2017;**31**:1-20

[6] Pahnehkolaei SMA, Alfi A, Machado JAT. Uniform stability of fractional order leaky integrator echo state neural network with multiple time delays. Information Sciences. 2017; **418-419**:703-716

[7] Guo Y, Wang F, Chen B, et al. Robust echo state networks based on correntropy induced loss function. Neurocomputing. 2017;**267**(6):295-303

[8] Lun SX, Yao XS, Qi HY, et al. A novel model of leaky integrator echo state network for time-series prediction. Neurocomputing. 2015;**159**(1):58-66

[9] Li D, Min H, Wang J. Chaotic time series prediction based on a novel robust echo state network. IEEE Transactions on Neural Networks and Learning Systems. 2012;**23**(5):787-799

[10] Schrauwen B, D'Haene M, Verstraeten D, et al. Compact hardware liquid state machines on FPGA for real-time speech recognition. Neural Networks the Official Journal of the International Neural Network Society. 2008;**21**(2–3):511

[11] Alomar ML, Canals V, Martinez-Moll V, et al. Low-cost hardware implementation of reservoir computers. In: International Workshop on Power and Timing Modeling, Optimization and Simulation. IEEE; 2014. pp. 1-5

[12] Jaeger H. Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the Echo State Network Approach - First revision. 2002. 7 p

[13] Sun X, Li T, Li Q, et al. Deep belief echo-state network and its application to time series prediction. Knowledge-Based Systems. 2017;**130**(15):17-29

[14] Yi Y, Liao Y, Wang B, et al. FPGA based spike-time dependent encoder and reservoir design in neuromorphic computing processors. Microprocessors and Microsystems. 2016;**46**(PB):175-183

[15] Jaeger H, Haas H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science. 2004; **304**(5667):78-80

[16] Kulkarni MS, Teuscher C. Memristor-based reservoir computing. In: IEEE/ACM International Symposium on Nanoscale Architectures. ACM; 2012. pp. 226-232

[17] Zhang L, Zhang X. MIMO channel estimation and equalization using three-layer neural networks with feedback. Journal of Tsinghua University Natural Science Edition (English Edition). 2007; **12**(6):658-662

[18] Potter C. RNN based MIMO channel prediction. Signal Processing. 2010; **90**(2):440-450

[19] Sarma KK, Mitra A. Modeling MIMO channels using a class of complex recurrent neural network architectures. AEU International Journal of Electronics and Communications. 2012;**66**(4): 322-331

[20] Routray G, Kanungo P. Rayleigh fading MIMO channel prediction using RNN with genetic algorithm. Communications in Computer and Information Science. 2011;**250**:21-29

[21] Cai H. MIMO-OFDM channel estimation based on neural network. Computer Engineering and Applications. 2011;**47**(34):1-4

[22] Lukoševičius M. A Practical Guide to Applying Echo State Networks. Springer; 2012

[23] Dorado-Moreno M, Cornejo-Bueno L, Gutiérrez PA, et al. Robust estimation of wind power ramp events with reservoir computing. Renewable Energy. 2017;**111**:428-437