Advances in Computer Science and its Applications (ACSA)
Vol. 2, No. 1, 2012, ISSN 2166-2924

www.worldsciencepublisher.org

N-bit Parity Implementation based on Binary Neural Networks with *O(n/2)* Neuron and *O(2n)* Connection

Ehsan Lotfi

Computer Department, Islamic Azad University, Torbat-e-Jam Branch, Torbat-e-Jam, Iran
Email: elotfi@ieee.org (Alternative Email: esilotf@gmail.com)

Abstract – In communication systems, n-bit parity problem (NPP) is widely used for error detection and correction. In this paper, an efficient architecture for hardware implementation of NPP is proposed. For this purpose, first we introduce extending single neuron that can be trained by perceptron learning rule to solve 2-bit and 3-bit parity problems. Then, we propose novel architecture of cascaded modular neural networks, based on the presented neuron, with *O(n/2)* neuron and with *O(2n)* connections to solve NPP. The main advantages of proposed parity networks are low number of neurons, connecting weights and inputs for each neuron.

Keywords – Perceptron; XOR; Neural Nets; Optimum Architecture

1. Introduction

The n-bit parity problem (NPP) is widely used in digital communication and transmission systems, especially in error detection and correction modules [1]. NPP is the extended *XOR* problem to the n dimensions and efficient hardware implementation of NPP allows communications systems to perform close to the channel capacity limit. There are various approaches for NPP implementation. One of these approaches is neural networks (NNs) implementation.

NNs approach focuses on following two features of NPP; first, the NPP shows high degree of data nonlinearity. And the second is curse of dimensionality, i.e., by increasing the number of inputs n, its computational complexity is significantly increased. In neural implementation of NPP, n concludes the number of neurons in the structure and for optimizing the architecture, the number of neurons as well as the number of connections must be reduced.

In the literatures, there are following four types of ANNs for NPP solving:

- NNs with complex-valued weights and complex-valued activation function (CWCA) [2-3].
- NNs with real weights and real monotonic activation functions (RWRA) [4].
- NNs with integer weights and periodic integer activation function (IWPA) [5].
- NNs with integer weights and integer threshold activation function (IWIA) [6-7].

Although, CWCA [8] and IWPA [9] can solve n-bit parity problem with *O(1)* nodes and RWRA needs just *O(n/2)* nodes in hidden layer to solve NPP [10]; Among these models, IWIA has important advantage that is facility in hardware implementation [6-7]. So many researchers have been tried to propose optimum architectures with IWIA which are summarized as follows.

Any Boolean function can be implemented in 2 hidden layers neural network with *O(2n/2)* nodes [6]. Kim and Park [7] proposed an architecture with *O(n)* hidden neurons to solve NPP. In this architecture, the inputs are fully connected with hidden neurons. So $O(n^2)$ connections is produced in the architecture. Liu et al. [11] solved NPP with *O(n/2)* hidden neurons. In their proposed architecture, the inputs are directly connected with output and the number of connections is $O(n^2/2)$. Wilamowski and Hunter [12] proposed several neural network architectures for NPP solving. These architectures and Arslanov's architecture [13] can solve the NPP with *O(logn)* neurons. The high number of connections and the real connecting weights are disadvantages of these architectures. Yang et al. [14] presented that *O(n)* neuron is required to solve NP. In this architecture [14], the number of connections with hidden neurons is $O(n^2)$. Franco and Cannas [15] introduced modular neural networks to solve NPP. This architecture is based on cascaded neural module solving XOR problem. The number of real connecting weights in this architecture is $O(n^2)$.

Here we introduce extending single neuron perceptron which can solve XOR and 3-bit parity problems. The proposed model increases the dimension of input vector and can be used in cascaded modular architecture to solve NPP. The proposed parity network needs just *O(n/2)* neurons and *O(2n)* connections to solve the problem.

2. Proposed Parity Network

Proposed parity network is constructed in two steps. At the first, extending single neuron perceptron is proposed to solve XOR problem. Then it is extended to

solve 3-bit parity. Finally the network is constructed by using the extended neuron.

2.1. Extending single neuron perceptron

Figure 1 shows the XOR problem where p_1 and p_2 are the inputs. In the Figure 1 the solid points remark the output "1" and the hollow points remark the output "0". It is clear that these points cannot be separated by single line. Figure 2 shows the extended *XOR* problem. As illustrated in the Figure 2, *XOR* points in three dimensions can be separated by single plane. The third dimension is produced by "AND" operator as follows:

$$p_3 = AND(p_1, p_2) \tag{1}$$

Eq. 1 is borrowed from the effect of "MAX" operator for real inputs in a computational algorithm that was presented in our recent work [16]. The "AND" operator for binary inputs and "MAX" operator for real inputs cause the linear separability of input data.

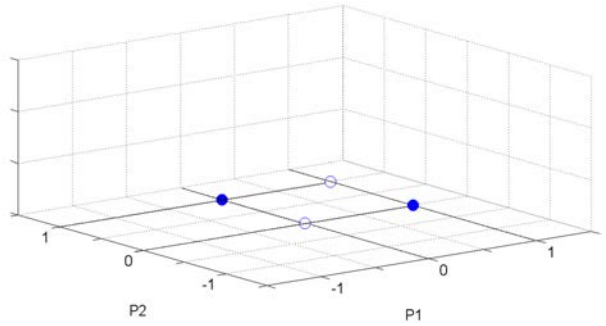

Figure 1. *XOR* problem in two dimentions which is not linearly separable

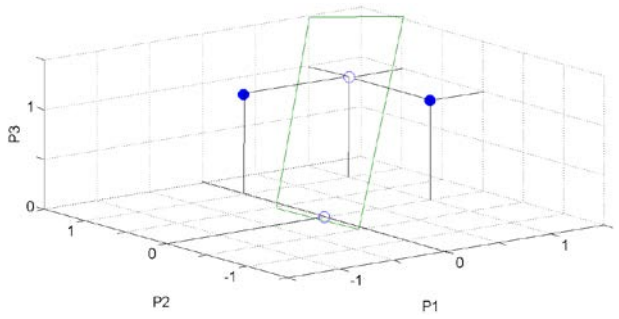

Figure 2. XOR problem in three dimentions which can be separated by single plane

Based on this extension, the single neuron perceptron can solve the XOR problem. In the Figure 3, the perceptron, presented as 3 inputs - single output, has input weights 1, 1, -3 and the bias value -1. The output of the neuron is calculated by following formula:

$$O = hardlim(p_1 \times w_1 + p_2 \times w_{23} + AND(p_1, p_2) \times w_3 + b) \tag{2}$$

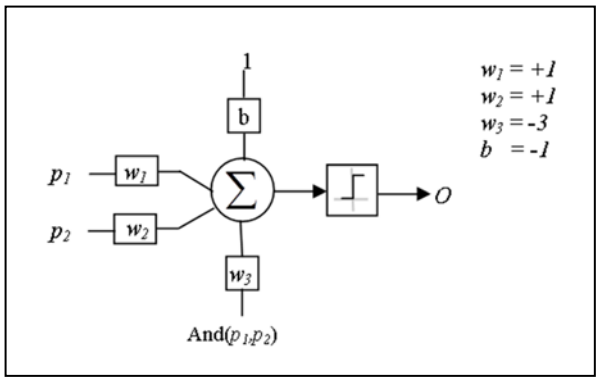

Figure 3. Two inputs extending neuron. The adjusted weights for XOR prblem are illustrated in the figure.

2.2. *3*-bit parity network

The perceptron presented in Figure 3 cannot approximate 3-bit parity function. Let us generalize the extending concept by using the "OR" operator as illustrated in Figure 4. The output of the extending perceptron in Figure 4 is calculated by following formula:

$$O = hardlim(p_1 \times w_1 + p_2 \times w_2 + p_3 \times w_3 + AND(p_1, p_2, p_3) \times w_4 + OR(p_1, p_2, p_3) \times w_5 + b) \tag{3}$$

Where w s are input weights and

$$hardlim(x) = \begin{cases} 1 & if\ (x \geq 0) \\ 0 & otherwise \end{cases} \tag{4}$$

The connecting weights can be adjusted by perceptron learning rule as follows:

$$w_i = w_i + (T - O)p_i \qquad i=1\ldots5 \tag{5}$$

Where T is related target with input pattern p_1, p_2, p_3 and O is output of neuron and p_4, p_5 calculated by following formulas are extended dimensions:

$$p_4 = AND(p_1, p_2, p_3) \tag{6}$$

$$p_5 = OR(p_1, p_2, p_3) \tag{7}$$

Actually the proposed extending perceptron, calculated by Eq. 3, has 5 inputs including p_1, p_2, p_3 (input pattern) and p_4, p_5 (extended connections). The learning rule illustrated by Eq. 5 keeps the weights in the integer space.

The proposed extending perceptron solves the 3-bit parity problem in the following manner; Consider the extending perceptron architecture illustrated in Figure 4 with the following weights: $w_1 = w_2 = w_3 = -2$ and $w_4 = 6$, $w_5 = 3$ and b = -1thus:

$$\hat{f}(1,1,1) = hardlim(w_1 \times 1 + w_2 \times 1 + w_3 \times 1 + w_4 \times AND(1,1,1) + w_5 \times OR(1,1,1) + b) = 1 \tag{8}$$

$$\hat{f}(1,1,0) = hardlim(w_1 \times 1 + w_2 \times 1 + w_3 \times 0 + w_4 \times AND(1,1,0) + w_5 \times OR(1,1,0) + b) = 0 \quad (9)$$

$$\hat{f}(1,0,1) = hardlim(w_1 \times 1 + w_2 \times 1 + w_3 \times 1 + w_4 \times AND(1,0,1) + w_5 \times OR(1,0,1) + b) = 0 \quad (10)$$

$$\hat{f}(0,1,1) = hardlim(w_1 \times 0 + w_2 \times 1 + w_3 \times 1 + w_4 \times AND(0,1,1) + w_5 \times OR(0,1,1) + b) = 0 \quad (11)$$

$$\hat{f}(1,0,0) = hardlim(w_1 \times 1 + w_2 \times 0 + w_3 \times 0 + w_4 \times AND(1,0,0) + w_5 \times OR(1,0,0) + b) = 1 \quad (12)$$

$$\hat{f}(0,1,0) = hardlim(w_1 \times 0 + w_2 \times 1 + w_3 \times 0 + w_4 \times AND(0,1,0) + w_5 \times OR(0,1,0) + b) = 1 \quad (13)$$

$$\hat{f}(0,0,1) = hardlim(w_1 \times 0 + w_2 \times 0 + w_3 \times 1 + w_4 \times AND(0,0,1) + w_5 \times OR(0,0,1) + b) = 1 \quad (14)$$

$$\hat{f}(0,0,0) = hardlim(w_1 \times 0 + w_2 \times 0 + w_3 \times 0 + w_4 \times AND(0,0,0) + w_5 \times OR(0,0,0) + b) = 0 \quad (15)$$

$\hat{f}$ based on Eq. 3 can approximate the 3-bit parity function. So the extending single neuron perceptron illustrated in Figure 4 can solve 3-bit parity problem.

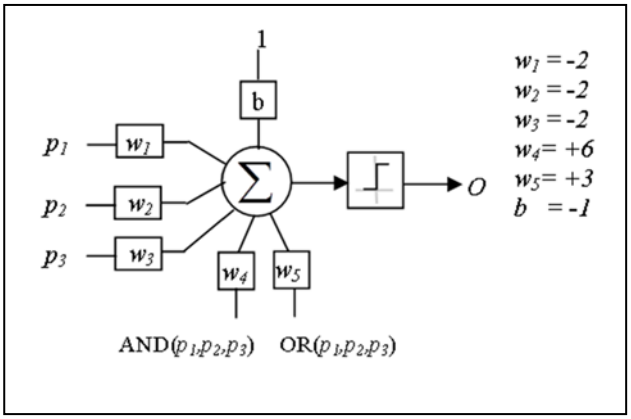

Figure 4. Single neuron extending perceptron with hardlim actication function, 3 input pattern, 2 input extended connections and bias. The adjusted weights for 3-bit parity problem are illustrated in the figure.

3. Cascaded Extending Perceptron

Figure 5 shows the proposed parity network for 8 bit-parity problem. Each hidden neuron in this network is the three inputs extending neuron illustrated in Figure 4 and output neuron is specified by two inputs extending neuron illustrated in Figure 3.

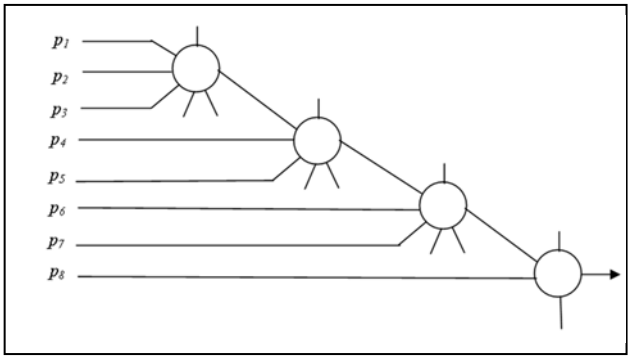

Figure 5. Parity network for 8-bit parity problem. For details of hidden neurons see Figure 3 and for details of output neuron see Figure 4.

Figure 6 shows the proposed parity network for 9-bit problem. Each neuron in this network is three inputs extending perceptron presented in Figure 3. NPP, when n is even, is constructed similar to Figure 5. And NPP, when n is odd, is constructed similar to Figure 6.

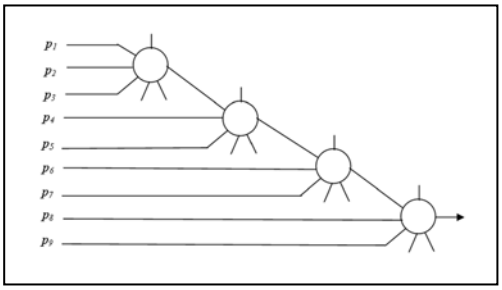

Figure 6. Parity network for 9-bit parity problem. For details of each neuron, see Fugire 4.

In the proposed parity network, the number of neurons for n-bit input is $\left\lfloor \frac{n}{2} \right\rfloor$. Thus the order of the number of nodes is *O(n/2)*. Also The number of "AND" and "OR" gates is (n-1) and the number of connecting weights is (2n+2). So the total number of nodes including gates belong to *O(n)* and the total number of connecting weights belong to *O(2n).*

4. Experimental Results on Convergence

Mean square error (MSE) is performance measure to evaluate the convergence of extending perceptron. The measure generally is expressed as follows:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(O_i - T_i)^2 \quad (16)$$

Where m is number of patterns and is equal to 2^n for NPP, *O* s are the outputs of the model and *T* s are desired outputs. Figure 7 shows the MSE of *XOR* learning. According to the Figure 7, tow inputs extending neuron just in 10 epochs learns the *XOR* problem. Figure 8 presents the MSE of 3-bit parity learning. As illustrated in Figure 8, extending single neuron perceptron just in 12

epochs learns the 3-bit parity problem. So extending neurons can learn the *XOR* and 3-bit parity problems quickly by perceptron learning rule. The adjusted weights presented in Figure 3 and Figure 4 are resulted from this learning and according to the Figure 5 and Figure 6, can be used in NPP solving.

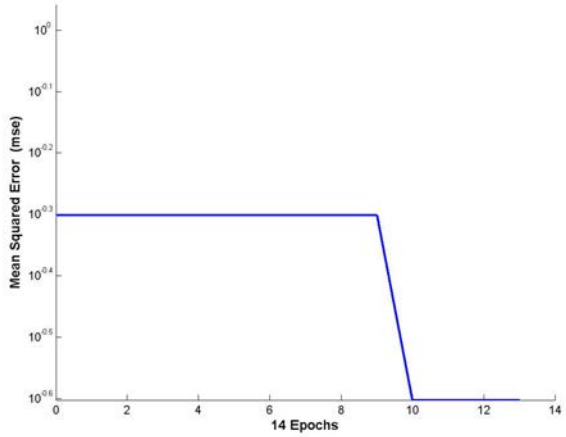

Figure 7. Performance measure in XOR learning epochs

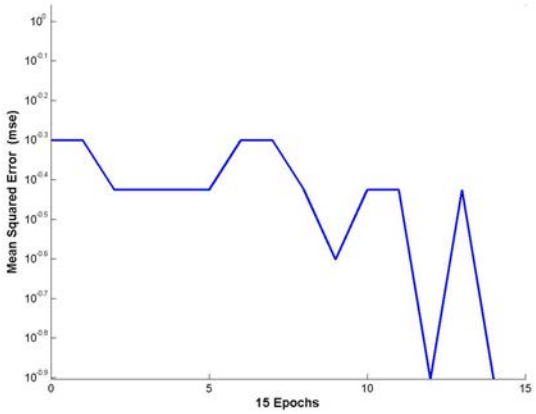

Figure 8. Performance measure in 3-bit parity learning epochs

2- Conclusion

According to the experimental results, the extending single neuron, proposed here, can learn the *XOR* and 3-bit parity problems by perceptron learning rule. Additionally, in this paper, a novel cascaded modular parity network is constructed based on proposed extending neuron to solve n-bit parity problems (NPP). The main advantage of presented parity networks is lower number of connecting weights. Table 1 shows a comparison between proposed parity network and other solutions. According to the Table 1, the lower number of connecting weights is obtained from the proposed solution. The network requires just *O(2n)* connection and *O(n/2)* neuron to solve NPP.

Table 1. Comparisons between various architecture of NPP

Solution[a]	Hidden neurons	Connecting weights
Kim and Park [7]	$O(n)$	$O(n^2)$
Liu et al. [11]	$O(n/2)$	$O(n^2/2)$
Wilamowski and Hunter [12]	$O(\log n)$	$O(n\log n)$
Arslanov et al. [13]	$O(\log n)$	$O(n^2)$
Yang et al. [14]	$O(n)$	$O(n^2)$
Franco and Cannas [15]	$O(n)$	$O(n\log n)$
Proposed parity network	$O(n/2)$	$O(2n)$

a. Solutions with threshold activation functions

References

[1] Rui Min, Jian-min He, Min He, Wei Ding, “Research of the Soft Decision Decoding Algorithm Based on Belief Propagation,” Advances in Information Technology and Management (AITM), Vol 1, No 4 (2012).

[2] I. Aizenberg, “Solving the XOR and Parity n Problems Using a Single Universal Binary Neuron,” Soft Computing, vol. 12, 2008, pp. 215-222.

[3] I. Aizenberg, N. Aizenberg and J. Vandewalle, ”Multi-valued and universal binary neurons: theory, learning, applications,” Kluwer Academic Publishers, Boston Dordrecht London, 2000.

[4] D. G. Stork and J. D. Allen,”How to solve the N-bit parity problem with two hidden units,” Neural Networks, vol. 5, pp. 923-926.

[5] Fangyue Chen, Guanrong Ron Chen, Guolong He, Xiubin Xu, and Qinbin He, “Universal Perceptron and DNA-Like Learning Algorithm for Binary Neural Networks: LSBF and PBF Implementations,” IEEE Trans Neural Netw, vol. 20, no. 10, Oct. 2009, pp. 1645-1658.

[6] K. Y. Siu, V. Roychowdhury, and T. Kailath, “Depthsize tradeofd for neural computation,” IEEE Trans. Comput., vol 40, pp. 1402-1412, Dec. 1991.

[7] Jung H. Kim and Sung-Kwon Park, “The Geometrical Learning of Binary Neural Networks” IEEE Trans Neural Netw, vol. 60, no. 1, Oct. 1995, pp. 237-248.

[8] I. Aizenberg, “Computational Intelligence, Theory and Application" (B. Reusch – Editor), Springer, Berlin, Heidelberg, New York, 2006, pp. 457-471.

[9] Fangyue Chen1,Wenhui Tang and Guanrong Chen, “Single-layer perceptron and dynamic neuron implementing linearly non-separable Boolean functions,” Int. J. Circ. Theor. Appl. 2009, vol. 37, pp. 433–451.

[10] M. E. Hohil, D. Liu, and S. H. Smith, “Solving the N-bit parity problem using neural networks,” Neural Networks, vol. 12, 1999, pp. 1321-1323.

[11] D. Liu, M.E. Hohil & S.H. Smith, “N-bit parity neural networks: new solutions based on linear programming,” Neurocomputing, vol. 48, pp.477-488, 2002.

[12] B. M. Wilamowski and D. Hunter, “Solving parity-n problems with feedforward neural network,” in Proc. IJCNN, Portland, OR, Jul. 2003, pp. 2546–2551.

[13] M. Z. Arslanov, D. U. Ashigaliev, andE. E. Ismail, “ N-bit parity ordered neural networks,” Neurocomputing vol. 48, 2002, pp. 1053–1056.

[14] Lu Y, Yang J, Wang Q, et al. “The upper bound of the minimal number of hidden neurons for the parity problem in binary neural networks,” Sci China Inf Sci, 2011, 54: doi: 10.1007/s11432-011-4405-6.

[15] L. Franco, SA Cannas, “Generalization properties of modular networks: implementing the parity function,” IEEE Trans Neural Netw 2001, vol. 12, pp. 1306–1313.

[16] E. Lotfi and M.R. Akbarzadeh-T., ”Supervised Brain Emotional Learning,” The 2012 International Joint Conference on Neural Networks(IJCNN), Australia , doi: 10.1109/IJCNN.2012.6252391, 2012, pp. 1-6.