2009 International Symposium on Nonlinear Theory and its Applications
NOLTA'09, Sapporo, Japan, October 18-21, 2009

Quinary Adder LOGO Neural Network Based on Mixed Radices

Hassan Amin Osseily, Ali Massoud Haidar*
Faculty of Engineering, Beirut Arab University, Beirut - Lebanon
Email : sasoha@yahoo.com, *ari@bau.edu.net

Abstract— **Our objective in this paper is to demonstrate an optimized method of the addition process in Quinary Logic (QL) adders, and which we will call the "mixed radices of Quinary / binary". Upon mixing radices (quinary / binary), we will be able to represent quinary numbers by using binary vectors with only two bits instead of three bits. Implementing this method, by using the Logic Oriented Neural Network (LOGO-NN), will enable us as well to reduce the number of needed elements and interconnections. The proposed adder will be compared with other techniques in order to evaluate its performance.**

Introduction

Rapid progress in the domain of neural networks and microelectronic technologies has increased enormously the need of development of high speed, efficient, and high computational engineering tasks. In this paper, a new neural networks approach has been proposed to implement a quinary arithmetic adder model. The LOGO-NN can perform several independent computations in parallel [1], by using a single network. The multiple-valued logic LOGO-NN [2] provides powerful computational capabilities for larger quantities of data. New LOGO-NN systems include associated mathematical tools which allow us to analyze and synthesize any logic model in a simple and systematical approach. The LOGO-NN is proposed in a way to form a complete system (completeness) that can realize any multiple-valued logic function [3]. It has been found that the mixed radices [4], [5] provide a convenient way to analyze, synthesize and minimize the multiple valued logic functions. Also, it has been proven that mixed radices LOGO-NN [4] allow us to reduce the number of elements and interconnections. In this paper, we will try to use mixed radices quinary /quaternary / binary to reduce the binary representation of quinary numbers, thus decreasing the number of elements and interconnections of quinary adder LOGO-NN.

I. NEURON MODEL

A. General Overview

The LOGO-NNs are composed of one neural type, and all the synapse's weights between neurons are taken as natural integers. These two characteristics make LOGO-NNs useful, simple to design and more realistic in comparison with that of [2]. The Galois field algebra [4] provides a convenient way to specify the structure of binary, ternary and quaternary. The LOGO-NN operators of Galois field along with the logic constants, form a finite field. The structure of k-valued logic LOGO-NN is defined as:

$$NNQ = (G, GF(k), f(Z)) \quad (1)$$

where,
G: Finite directed graph under the form:

$$G = (N, L, W) \quad (2)$$

where,
N: is the Set of nodes (neurons).
L: is the Set of links (connections).
W: is the Set of synapse's weights.
GF (k) = Galois field of k elements, defined as:

$$GF(k) = \{0, 1, 2 \ldots k-1\} \quad (3)$$

$$k \geq 2 \quad (4)$$

$f(Z)$: Output signal of processing elements (neuron)

$$f(Z) = \begin{cases} Z, & Z > 0 \\ 0, & Z \leq 0 \end{cases} \quad (5)$$

$$Z = \sum_{i=0}^{n} x_i w_i - \theta \quad (6)$$

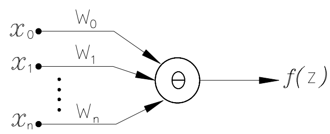

Figure 1: Processing element

where,
x_i: Input signals.
$x_i \in GF(k) = \{0, 1, 2 \ldots K-1\}$.
w_i: Multiplicative coefficient (weight) for x_i
i = 0, 1… n
θ: Threshold of the processing element.
$w_i, \theta \in \{\ldots, -2, -1, 0, 1, 2 \ldots\}$.

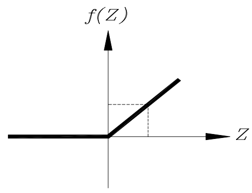

Figure 2: Linear transfer function

B. Galois field of 2 elements LOGO-NN

Any binary logic function can be represented by the familiar Galois field structures [4]. The flexibility of this modular algebra demonstrated above, is its suitability for the applications of LOGO-NN. The Galois field of 2-elements, for example, has a value of K = 2, thus GF (2) = {0, 1}. Where GF (2) is defined by the addition (⊕) and multiplication (•) functions, as given in table 1 below and as shown in figure 3.

⊕	0	1
0	0	1
1	1	0

•	0	1
0	0	0
1	0	1

Table 1: ⊕ and • Functions of GF (2).

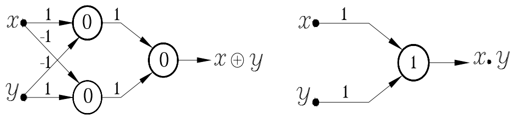

Figure 3: LOGO-NN of GF (2) functions.

C. *Basic Binary LOGO-Neural Networks*

The basic binary Logo Neural Networks are similar to those used in binary logic function such as the complement function, AND, OR, NOR, NAND…etc.

Complement function:

The complement function is defined by (7) and Table 2, where its LOGO-NN operator is designed as shown in Fig. 4.

$$x = 1 - \bar{x} \qquad (7)$$

TABLE 2: COMPLEMENT FUNCTIONS

x	$\bar{x}$
0	1
1	0

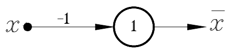

Figure 4: LOGO-NN of complement function

GF (2) Multiplication of n-input signals:

The LOGO-NN of GF (2) multiplication of n-input signals (Fig. 5) is designed for:

$$w_0 = w_1 = w_2 = \dots = w_n = 1 \text{ and } \theta = n$$

Then:

$$f(Z) = x_0 \bullet x_1 \bullet x_2 \bullet \dots \bullet x_n \qquad (8)$$

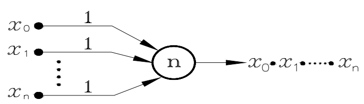

Figure 5: LOGO-NN for GF (2) multiplication of n-input

OR function:

The OR function is defined as given in Table 3 and as shown in figure 6.

+	0 1
0	0 1
1	1 1

TABLE 3: OR FUNCTIONS

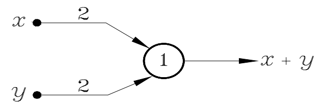

Figure 6: LOGO-NN of OR function

Minimization rule of LOGO Neural Networks:

The LOGO neural networks of Figure 7, shows a simple example for reduction rule that can be used to minimize LOGO-NN of the expression (f= x • y •$\bar{z}$)

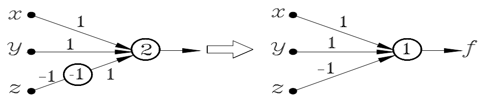

Figure 7: Minimized Network.

II. QUINARY LOGIC ADDER BASED ON MIXED RADICES AND ITS LOGO-NN IMPLEMENTATION

Quinary Logic Adderer:

The quinary logic addition process of two quinary input variables is defined as given in table 4, where S is the sum (S = X • Y) and C is the carry of S.

TABLE 9: REPRESNTATION OF SUM AND CARRY

Y	X	S	C
0	0	0	0
0	1	1	0
0	2	2	0
0	3	3	0
0	4	4	0
1	0	1	0
1	1	2	0
1	2	3	0
1	3	4	0
1	4	0	1
2	0	2	0
2	1	3	0
2	2	4	0
2	3	0	1
2	4	1	1
3	0	3	0
3	1	4	0
3	2	0	1
3	3	1	1
3	4	2	1
4	0	4	0
4	1	0	1
4	2	1	1
4	3	2	1
4	4	3	1

X, Y and S belong to the quinary set {0, 1, 2, 3, and 4}. To represent these variables under a binary form, we need three bits for each. Then we will obtain 64 different binary combinations between X and Y where 39 of them will be dropped/unused.

Let: Y=(Y3,Y2,Y1), X=(X3,X2,X1), S=(S3,S2,S1) and C=(c2,c1)

In order to reduce the calculation complexity, we will try to represent the input variables by two binary bits only instead of three bits and hence the problem will become similar to the quaternary issue [4] ,thus we will have only 16 binary combinations and this will lead us to minimize the expressions of the functions of S and C. To achieve this objective, we suppose the following methodology:

- We have noticed from table 4 that C=0 when: X=0 or Y=0 and S=X or S=Y, and that for $X=\overline{Y1}$ then C=1 and S=0. This will allow us to remove all these cases from table 4 because the result here could be predicted. Hence the values of X, Y and S belong now to the set {1, 2, 3, 4}.

TABLE 8: X, Y, AND S FOR X≠0, Y≠0

Y	X	S	C
1	1	2	0
1	2	3	0
1	3	4	0
2	1	3	0
2	2	4	0
2	4	1	1
3	1	4	0
3	3	1	1
3	4	2	1
4	2	1	1
4	3	2	1
4	4	3	1

- The next step is to subtract “1” from the digits of X and Y. Thus, the set of numbers becomes the same of the quaternary one {0,1,2,3}.
- After the subtraction , we could then represent the numbers 0,1,2 and 3 by two bits binary vectors, that is to say 0=(0,0), 1=(0,1), 2=(1,0) and 3=(1,1).Hence, and after we apply the above procedures, we obtain a reduced table which is shown in table 8. The equations of *y*, *x* , *S* and *C* are as follows:

$$y=Y-1=(y2,\ y1) \qquad (23)$$

$$x=X-1=(x2,\ x1) \qquad (24)$$

$$S0=(s3,\ s2,\ s1) \qquad (25)$$

$$C=(c1) \qquad (26)$$

TABLE 9: REPRESNTATION OF X AND Y BY 0,1

y2,y1	x2,x1	S3,S2,S1	C
0,0	0,0	0,1,0	0
0,0	0,1	0,1,1	0
0,0	1,0	1,0,0	0
0,1	0,0	0,1,1	0
0,1	0,1	1,0,0	0
0,1	1,1	0,0,1	1
1,0	0,0	1,0,0	0
1,0	1,0	0,0,1	1
1,0	1,1	0,1,0	1
1,1	0,1	0,0,1	1
1,1	1,0	0,1,0	1
1,1	1,1	0,1,1	1

Consequently, we obtain the traditional functions s1, s2, c1 with four variables x1, x2, y1, and y2. To find the optimized functions, it could be done easily by using karnaugh map [6], The following expressions will be obtained:

$$S01= x1.x2.\ y2 + x2.y1.y2+\overline{x1}.x2\ \overline{y1}.\overline{y2}+ x1.\overline{x2}\ \overline{y1}.\overline{y2} + \overline{x1}.\overline{x2}\ y1.\overline{y2} \qquad (27)$$

$$S02= x1.x2.\ y1 + x1.y1.y2 + \overline{x1}.\overline{x2}\ \overline{y2} + \overline{x2}.\overline{y1}\ \overline{y2} \qquad (28)$$

$$S03=\overline{S1 + S2} \qquad (29)$$

$$C0= x2.y2 + x1.y1.y2 + x1.x2.y1 \qquad (30)$$

The block diagram in figure 8 shows the major units involved in the structure of the quinary adder. This block diagram is composed of three units. The input unit is the quinary to binary converter which is designed by LOGO-NN to convert directly any quinary number to a binary one with two bits only. Many methods were proposed [7], [8] to design radix converters. The second unit is the LOGO-NN adder which composes the heart or the main unit.

The LOGIC UNIT is the output unit with binary coded quinary where we get the final sum and carries resulting from the addition process of the two quinary numbers X and Y.

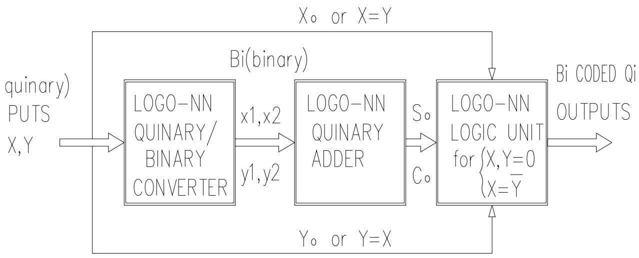

Figure 8: Block Diagram for Quinary / Binary Adder.

The LOGO-NN of the three units is as shown in the figures 9, 10 and 11 respectively.

In figure 9, the LOGO-NN quinary to binary converter is designed to give for certain quinary inputs (X and Y), the corresponding binary vectors with two bits (x1, x2) and (y1, y2). First, we have a rotary switch that select one quinary digit as input. For the input 0, we put an inverter to get the logical "0" when we activate the input by a logical "1". Hence, we deal with the remaining 4 quinary inputs (1, 2, 3, and 4) as if they were quaternary inputs and which, in their turn, need two binary bits only for representation.

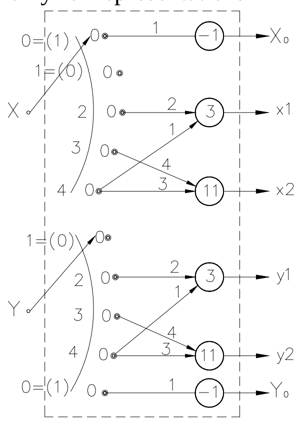

Figure 9: Quinary to Binary converter LOGO-NN

Figure 10 shows the main unit of the adder with mixed radices quaternary to binary coded quinary. The LOGO-NN implementation for equations 15,16,17,18 and 19 are given as per paragraph II (Neuron Model).

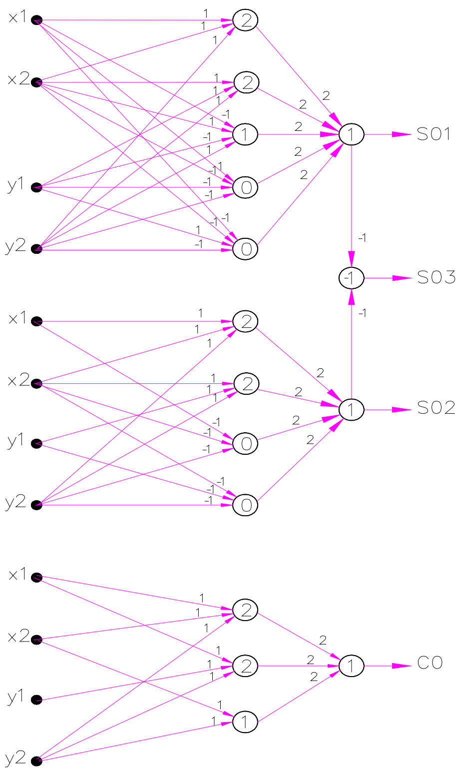

Figure 10: Multiplier Unit LOGO-NN.

III. Conclusion

We presented in this paper, the development of a new technique concerning the quinary adder LOGO-NN through the use of mixed radices (quinary / quaternary / binary) in order to simplify and minimize the used elements and to increase the performance of quinary adders to the maximum. The advantages of mixing the radices are the simplicity of binary design, and the availability of binary components. The proposed algorithm makes it possible to represent quinary as binary with two variables only. While in general it needs to be represented with three bits for each quinary digit (quits). For evaluation purposes, simulations were done under these circumstances:

1. Interesting results were obtained upon applying a simulation on MATLAB R2007A using a core 2 duo processor of frequency 1.73 GHz. where we noticed that it takes much less time to add two numbers using the mixed radices algorithm, in comparison with that of classical binary addition which is already embedded in the computer processor.

2. In comparison with other adders' techniques [2], the advantages of this method can be summarized by:

- The proposed adder is composed of one neuron type, while that of [2] is composed of three neuron types.
- The proposed adder performs the complete operation in a single LOGO neural network, while that of [2] is decomposed into three sub-circuits which are controlled by an individual circuit.
- Integers are representing the synapse's weights and thresholds of neurons while that of reference [2] are non integer numbers.

References

[1] Joy Rogers, "Object-Oriented Neural Networks in C++". Academic Press 1997, PP. 83-131, Alabama, USA.

[2] Hu, C.J "Design of a 4-Valued Digital Multiplier Using an Artificial Heterogeneous Two Layered Neural Network", ISMVL Proc., PP. 84-87, 1992.

[3] Goerge Epstein, "Multiple–Valued Logic Design: an introduction" . PP.12-51University of North Carolina, computer science department, USA. 1993.

[4] A. M. Haider, "Analysis and Design of Multiple valued logic systems" Saitama University, Ph.D. thesis, PP. 83-112, Japan 1995.

[5] C. H. Huang, "A fully parallel mixed-radix conversion algorithm for residue number applications," *IEEE Trans. Comput.*, vol. 32, pp. 398-402, 1983.

[6] Albert Paul Malvino and Jerald A. Brown, "Digital Computer Electronics" third edition, Glencoe/McGraw-Hill publishing Company Limited, USA, pp.71-781999.

[7] T. Sasao, "Radix converters: Complexity and implementation by LUT cascades," *35th International Symposium on Multiple-Valued Logic*, Calgary, Canada, May 19-21, 2005, pp.256-263.

[8] Haidar, A., Osseily, H., Shirahama, H., and E. Nassar "Quinary Coded Decimal Conversion Techniques", Non Linear Theory Application Symposium NOLTA 2005, Belgium-Bruge. pp.70-73.