MVL multiplier logo neural network based on mixed radices

Hassan Amin Osseily [a], Ali Massoud Haidar [a] and Magdy Abd El-Azim Ahmed [b]
[a] *Faculty of Engg. Beirut Arab University, Beiru, Lebanon*
e-mail: [a] sasoha@yahoo.com, [a] ari@bau.edu.net
[b] *Computer and Systems Engg. Dept., Faculty of Engg., Alexandria University, Alexandria, Egypt*

The main purpose of this paper is to optimize the multiplication process in Multiple Valued Logic (MVL) multipliers by using a method called "mixed radices of MVL/binary". An introduction of ternary multipliers is mentioned only as a preface for quinary multipliers. The mixing of radices (quinary/binary) will allow to represent quinary numbers by binary vectors with two bits only instead of three bits. The implementation of this method by using the Logic Oriented Neural Network (LOGO-NN) will also enable us to reduce the number of elements and interconnections. For evaluation purposes, we will compare the proposed multiplier with other techniques.

ان الهدف الأساسي من هذا البحث هوتصميم ضارب متعدد القيم في الشبكات العصبية بالاعتماد على الجذور المختلطة للأنظمة متعددة المنطق مع النظام الثنائي وإيجاد معادلات هذا الضارب وتصميم دوائره المنطقية بواسطة الشبكات العصبية .في مقدمة البحث سنقوم بإجراء الدراسة على النظام الثلاثي ومن ثم الانتقال الى النظام الخماسي .وقد تبين بأن المزج بين النظام الخماسي مثلاً والنظام الثنائي يؤدي الى تقليل عدد الخانات الثنائية اللازمة لتمثيل الأعداد الخماسية من ثلاث خانات الى خانتين فقط ومن ذلك تبين ان تطبيق معادلات الضارب متعدد القيم باستخدام الشبكات العصبية يؤدي الى تقليل عدد العناصر التي يحتاجها وكذلك عدد التقاطعات في الدارة وبالتأكيد يؤدي الى تقليل عمليات المعالجة اللازمة لإنجاز عملية الضرب بين عددين .من أجل إجراء عملية تقييم لهذا الضارب فإننا سوف نقارنه مع تقنيات وتصاميم لباحثين آخرين.

Keywords: Logo neural network, Mixed radices, Multiple-valued logic, MVL multiplier

1. Introduction

Recent advances in neural sciences and microelectronic technologies increased greatly the challenges in the domain of development of high speed, efficient, complicated, and high computational engineering tasks. In this paper, a new approach of neural networks, to implement a quinary arithmetic multiplier model, has been proposed. The LOGO-NN is able to perform several independent computations in parallel [1] by a single network. The multiple-valued logic LOGO-NN [2] provides powerful computation for large quantities of data. The new Logic Oriented Neural Network (LOGO-NN) system is accompanied by mathematical tools which will 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 the mixed radices LOGO-NN [4] allow us to reduce the number of elements and interconnections. In this paper, we will use mixed radices quinary /quaternary / binary to reduce the binary representation of quinary numbers hence to reduce the number of elements and interconnections of quinary multiplier LOGO-NN.

2. Neuron model

2.1. General overview

The LOGO-NNs are composed of one neural type and all the synapse's weights between neurons are 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 is defined as:

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

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

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

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

$$f(Z) = \begin{cases} Z, Z>0 \\ 0, Z\leq 0 \end{cases}, \quad (4)$$

$$Z = \sum_{i=0}^{n} x_i w_i - \theta, \quad (5)$$

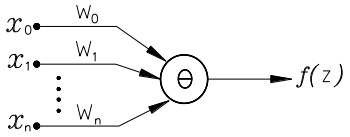

Fig. 1. Processing element.

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 $= 0, 1, \ldots, n$
θ Threshold of the processing element
$w_i, \theta \in$ $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$

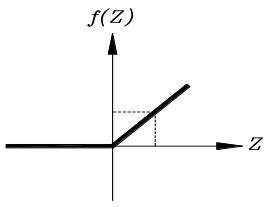

Fig. 2. Linear transfer function.

2.2. 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 is defined as K = 2, then GF (2) = {0, 1}. Where GF(2) is defined by addition (⊕) and multiplication (•) functions, as given in the table 1 below and as shown in fig. 3.

Table 1
⊕ and • functions of GF(2)

⊕	0	1
0	0	1
1	1	0

•	0	1
0	0	0
1	0	1

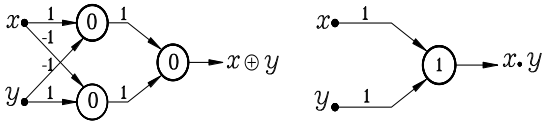

Fig. 3. LOGO-NN of GF(2) functions.

2.3. Basic binary LOGO-neural networks

The basic binary Logo Neural Networks are the same as in binary logic function such as complement, AND, OR, NOR, NAND.

Complement function

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

$$\bar{x} = 1 - x \quad (6)$$

Table 2
Complement functions

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

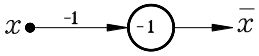

Fig. 4. LOGO-NN of complement function.

GF(2) Multiplication of n-input signals

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

$w_0 = w_1 = w_2 = \ldots = w_n = 1$ and $\theta = n$

Then:

$$f(Z) = x_0.\, x_1.\, x_2 \ldots x_n \quad (7)$$

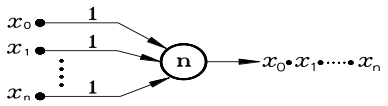

Fig. 5. LOGO-NN for GF(2) multiplication of n-input.

OR function

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

Table 3
Or functions

+	0	1
0	0	1
1	1	1

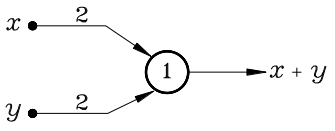

Fig. 6. LOGO-NN of OR function.

Minimization rule of LOGO neural networks

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

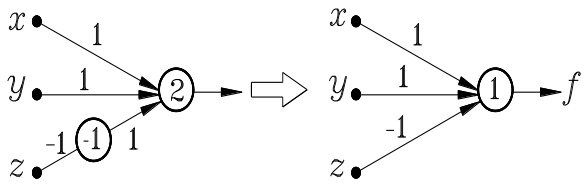

Fig. 7. Minimized network.

3. Ternary multiplier based on mixed radices

3.1. Ternay logic multiplier

The ternary logic multiplication process of two ternary input variables is defined as given in table 4, where M is the multiplier (M = X.Y) and C is the carry of M.

Table 4
Ternary multiplication table

Y	X	M	C
0	0	0	0
0	1	0	0
0	2	0	0
1	0	0	0
1	1	1	0
1	2	2	0
2	0	0	0
2	1	2	0
2	2	1	1

X, Y and M belong to ternary set {0,1,2}. To represent these variables in binary, we need two bits for each. Then we will obtain 16 different binary combinations between X and Y where 7 of them will be dropped or unused.

Let: Y = (Y2, Y1), X=(X2,X1), M=(M2,M1)
and C= (c)

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

- We have noticed from table 4 that M=C=0 when X=0 or Y=0 and this will enable us to remove these cases from table 4 because the result here could be predicted. Hence the values of X, Y and M belong now to the set {1,2}.
- The next step is to subtract "1" from the digits of X, Y and M, then the set of numbers become the same of the binary one {0,1}.

• After the subtraction, we could then represent the numbers 0, 1 by one bit binary vectors. Hence, and after we apply the above procedures, we obtain a reduced table which is shown in table 5. The equations of *y, x, m* and *C* are as follows,

$$y = Y\text{-}1 \tag{8}$$

$$x = X\text{-}1 \tag{9}$$

$$m = M\text{-}1 \tag{10}$$

$$C = c \tag{11}$$

Table 5
Subtraction of 1 from
X, Y, and M for X≠0, Y≠0

y	x	m	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Now we have a traditional functions *m*, and *c* with two variables *x, y*. To find the optimized functions, it could be done by using karnaugh map [6], where we have obtained the following

$$m = x \; XOR \; y = x.\ \bar{y} + \bar{x}\ .y \tag{12}$$

$$c = (x.\ y) \tag{13}$$

But actually we have to find the real multiplier *M*(*M2,M1*),where *M*= (*m+1*). Table 5 becomes, after the addition of '1' as shown below in table 6.

$$M = m+1 = (x \; XOR \; y) + 1 \tag{14}$$

Table 6
Addition of 1 for *m* (*M=m+1*)

y	*x*	(*M01,M02*)	*c*
0	0	0,1	0
0	1	1,0	0
1	0	1,0	0
1	1	0,1	1

By reference to the results obtained in the previous two tables 5 and 6, we notice that the equations of *M* and *C*:

$$M01 = x \; X \; OR \; y = x\bar{y} + \bar{x}\,y \tag{15}$$

$$M02 = \overline{M01} \tag{16}$$

$$C0 = c = x.y \tag{17}$$

But in the cases where *X=0* or *Y=0*, all the equations of *M* and *C* should equal to zero. Therefore, the final equations of *M* should be ANDed by *X.Y*
Where

$$X,Y = \begin{cases} 0 & if\ X = 0 \text{ or } Y = 0 \\ 1 & \text{otherwise} \end{cases} \tag{18}$$

Hence, the final equations of the multiplier become:

$$M1 = M01.X.Y \tag{19}$$

$$M2 = M02.X.Y \tag{20}$$

$$C1 = C01.X.Y \tag{21}$$

The final equations could be easily implemented by the basic binary LOGO-NN. The main purpose for analyzing ternary multiplier based on mixed radices is to introduce the quinary multiplier as described in the next paragraph.

4. Quinary logic multiplier based on mixed radices and its logo-nn implementation

4.1. Quinary logic multiplier

The quinary logic multiplication process of two quinary input variables is defined as given in table 7, where M is the multiplier (M= X.Y) and C is the carry of M.

X, Y and M belong to quinary set {0,1,2,3,4}. To represent these variables in binary, 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 or unused.

Let:Y=(Y3,Y2,Y1),X=(X3,X2,X1),M=(M3,M2,M1) and C=(c2,c1).

Table 7
Quinary multiplication table

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

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 M and C. To achieve this objective, we suppose the following methodology:

- We have noticed from table 4 that M=C=0 when X=0 or Y=0 and this will enable us to remove these cases from table 4 because the result could be predicted. Hence the values of X, Y and M belong now to the set {1,2,3,4}.
- The next step is to subtract "1" from the digits of X, Y and M, then the set of numbers becomes the same of the quaternary one {0,1,2,3}.

After the subtraction , we 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 procedure, we obtain a reduced table which is shown in table 8. The equations of *y, x, m* and *C* are as follows,

$$y=Y-1=(y2,y1) \quad (22)$$

$$x=X-1=(x2,x1) \quad (23)$$

$$m=M-1=(m2,m1) \quad (24)$$

$$C=(c2,c1) \quad (25)$$

Table 8
Subtraction of 1 from X, Y, and M for X≠0, Y≠0

y2,y1	x2,x1	m2,m1	c2,c1
0,0	0,0	0,0	0,0
0,0	0,1	0,1	0,0
0,0	1,0	1,0	0,0
0,0	1,1	1,1	0,0
0,1	0,0	0,1	0,0
0,1	0,1	1,1	0,0
0,1	1,0	0,0	0,1
0,1	1,1	1,0	0,1
1,0	0,0	1,0	0,0
1,0	0,1	0,0	0,1
1,0	1,0	1,1	0,1
1,0	1,1	0,1	1,0
1,1	0,0	1,1	0,0
1,1	0,1	1,0	0,1
1,1	1,0	0,1	1,0
1,1	1,1	0,0	1,1

Now we have a traditional functions m1, m2, c1 and c2 with four variables x1, x2, y1, and y2. The optimized functions can be easily done by using karnaugh map [6], where we have obtained the followings.

$$m1= x1.\overline{y1}\,\overline{y2} + \overline{x2}.y1.\overline{y2} + \overline{x1}.y1.y2 + x2.\overline{y1}.y2 \quad (26)$$

$$m2 = x2.\overline{y1}.\overline{y2} + x1.y1.\overline{y2} + \overline{x2}.y1.y2 = \overline{x1}.\overline{y1}.y2 \quad (27)$$

$$c1 = x2.y1\,\overline{y2} + x1.y1.y2 + x1.\overline{x2}\,y2 + \overline{x1}.x2.\overline{y1}.y2 \quad (28)$$

$$c2= x2.y1.y2 + x1.x2.y2 \quad (29)$$

But actually we have to find the real multiplier M(M3,M2,M1),where M= (m+1). Table 5 becomes, after the addition of '1' as shown below in table 9.

Table 9
Addition of 1 for m (M=m+1)

y2,y1	x2,x1	M03,M02,M01	C2,C1
0,0	0,0	0,0,1	0,0
0,0	0,1	0,1,0	0,0
0,0	1,0	0,1,1	0,0
0,0	1,1	1,0,0	0,0
0,1	0,0	0,1,0	0,0
0,1	0,1	1,0,0	0,0
0,1	1,0	0,0,1	0,1
0,1	1,1	0,1,1	0,1
1,0	0,0	0,1,1	0,0
1,0	0,1	0,0,1	0,1
1,0	1,0	1,0,0	0,1
1,0	1,1	0,1,0	1,0
1,1	0,0	1,0,0	0,0
1,1	0,1	0,1,1	0,1
1,1	1,0	0,1,0	1,0
1,1	1,1	0,0,1	1,1

By reference to the results obtained in the previous two tables 8 and 9, we notice that the equations of M:

$$M01= \overline{m1} \quad (30)$$

$$M02=m1 \text{ XOR } m2=m1.\overline{m2}.\overline{m1}.m2 \quad (31)$$

$$M03= (m1.m2) \quad (32)$$

But in the cases where X=0 or Y=0, all the equations of M and C should equal to zero. Therefore, the final equations of M should be ANDed by X.Y
Where

$$X,Y = \begin{cases} 0 & if\ X = 0 \text{ or } Y = 0 \\ 1 & \text{otherwise} \end{cases}$$

Hence, the final equations of multiplier become:

$$M1= M01.X.Y \quad (33)$$

$$M2=M02.X.Y \quad (34)$$

$$M3=M03\ X.Y \quad (35)$$

$$C1=C01.X.Y \quad (36)$$

$$C2=C02.X.Y \quad (37)$$

The block diagram in fig. 8 shows the major units involved in the structure of the multiplier. 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 numbers to binary with two bits only. Many methods were proposed [7, 8] to design radix converters. The second unit is the LOGO-NN multiplier which represents the heart or the main unit.

The ANDING with X0Y0 at the output unit with binary coded quinary where we get the final results for the multiplier and carries of the multiplication process of the two quinary numbers X and Y.

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

In fig. 9, the LOGO-NN quinary to binary converter is designed to give for quinary inputs (X and Y) 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 0 logic when we activate the input by 1 logic. Hence, we deal with the remaining 4 quinary iputs (1, 2, 3, 4) are considered as quaternary inputs that need two binary bits only for representation.

Fig. 10 shows the LOGO-NN implementation of the main unit of the multiplier with mixed radices quaternary to binary coded quinary (LOGO-NN of the eqs. (27-32 and 33).

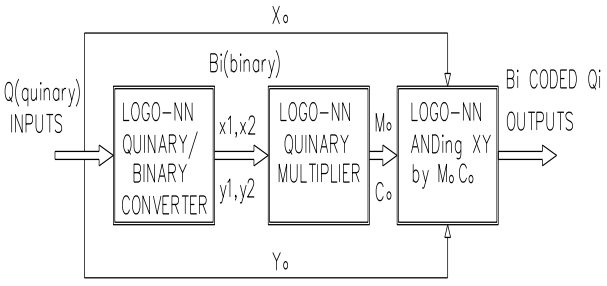

Fig. 8. Block diagram for quinary / binary multiplier.

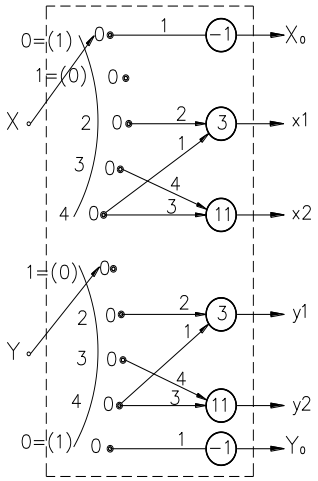

Fig. 9. Quinary to Binary converter LOGO-NN.

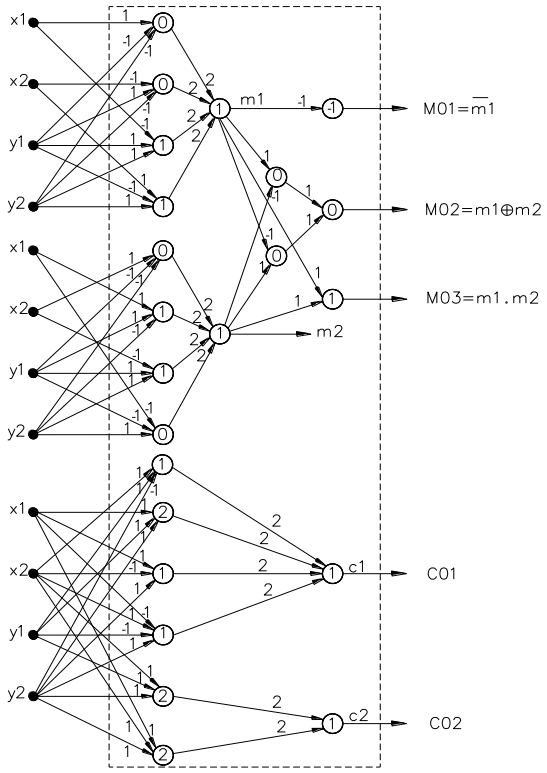

Fig. 10. Multiplier Unit LOGO-NN.

In fig. 11, we get the final multiplier and carries through the LOGO-NN of the ANDing unit with X0Y0 (LOGO-NN of the eqs. 35-38 and 39).

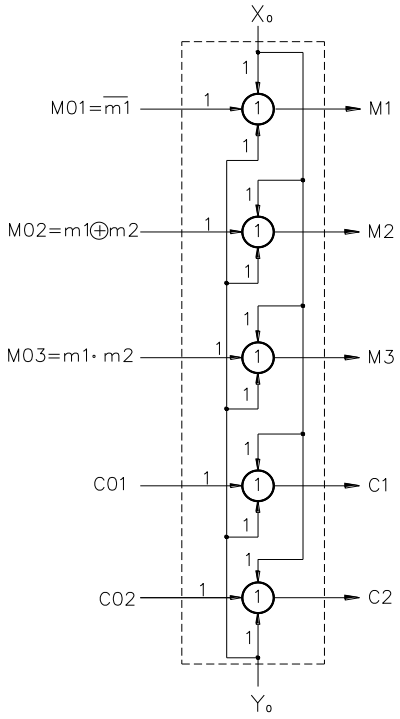

Fig. 11. ANDing (X0Y0) by (M0, C0) LOGO-NN.

5. Conclusions

This paper presents the development of a new technique concerning the quinary multiplier LOGO-NN through the use of mixed radices (quinary / quaternary / binary) in order to simplify and minimize as possible the used elements and to increase the performance of quinary multiplier to the maximum. The advantages of mixing the radices are the simplicity of binary design, and the availability of binary components. The algorithm here makes the representation of the quinary as binary with two variables only, possible. But, in general, it needs to be represented with three bits for each quinary digit (quits). For evaluation, the advantages of the proposed quinary LOGO neural network multiplier are the followings:

1. Interesting results were obtained upon doing a simulation on MATLAB R2007A using a core 2 duo processor of frequency 1.73 GHz, where we found that it takes about 8 microseconds to multiply two numbers (4 * 3 for instance) using the mixed radices algorithm, while it takes about 80 microseconds using a classical binary

multiplication which is already embedded in the computer processor.

2. In comparison with other multipliers techniques [2], the advantages are:

- The proposed multiplier is composed of one neuron type, while that of [2] is composed of three neuron types.
- The proposed multiplier 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 ref. [2] are non integer numbers.

References

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

[2] C.J. Hu, "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. 12-51 University of North Carolina, Computer Science Department, USA (1993).

[4] M. Haider, "Analysis and Design of Multiple Valued Logic Systems" Saitama University, Ph.D. Thesis, pp. 83-112, Japan (1995).

[5] 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. 71-78 (1999).

[7] T. Sasao, "Radix Converters: Complexity and Implementation by LUT Cascades", 35th International Symposium on Multiple-Valued Logic, Calgary, Canada, May 19-21, pp. 256-263 (2005).

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

Received December, 13 2008
Accepted May 28, 2009