World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:5, No:8, 2011

Integrating Fast Karnough Map and Modular Neural Networks for Simplification and Realization of Complex Boolean Functions

Hazem M. El-Bakry

***Abstract*—** In this paper a new fast simplification method is presented. Such method realizes Karnough map with large number of variables. In order to accelerate the operation of the proposed method, a new approach for fast detection of group of ones is presented. Such approach implemented in the frequency domain. The search operation relies on performing cross correlation in the frequency domain rather than time one. It is proved mathematically and practically that the number of computation steps required for the presented method is less than that needed by conventional cross correlation. Simulation results using MATLAB confirm the theoretical computations. Furthermore, a powerful solution for realization of complex functions is given. The simplified functions are implemented by using a new desigen for neural networks. Neural networks are used because they are fault tolerance and as a result they can recognize signals even with noise or distortion. This is very useful for logic functions used in data and computer communications. Moreover, the implemented functions are realized with minimum amount of components. This is done by using modular neural nets (MNNs) that divide the input space into several homogenous regions. Such approach is applied to implement XOR function, 16 logic functions on one bit level, and 2-bit digital multiplier. Compared to previous non- modular designs, a clear reduction in the order of computations and hardware requirements is achieved.

***Keywords*—**Boolean Functions, Simplification, Karnough Map, Implementation of Logic Functions, Modular Neural Networks.

I. INTRODUCTION

MINIMIZATION of the Boolean expression is very important. The purpose of simplification of Boolean functions is to reduce the number of gates in a logic circuit. By simplifying the logic function, the original number of digital components (gates) required to implement digital circuits can be reduced. Less number of logic gates means less power consumption, sometimes the circuit works faster and also when number of gates is reduced, cost also comes down. Therefore, by reducing the number of gates, the chip size and the cost will be reduced and the computing speed will be increased [1-35]. There are many ways to simplify a logic design, such as algebraic simplification, Karnough maps, Tabulation Method and Diagrammatic technique using 'Venn-like diagram'. Karnough map has the advantage that it is simple to realize and easy to implement. The Karnough map technique was proposed by M. Karnaugh [9]. Later Quine and McCluskey reported tabular algorithmic techniques for the optimal Boolean function minimization [10,11]. Almost all techniques have been embedded into many computer aided design packages and in all the logic design university textbooks [1-37]. K-map is a graphical representation of a truth table using Gray code order. It is suitable for elimination by grouping redundant terms in a Boolean expression. By optimizing the algorithm it is possible to simplify entirely a given Boolean expression. Unfortunately almost all the techniques along with the Espresso technique [14] do not always guarantee optimal solutions.

The main objective of this paper is to solve the problem of minimizing Boolean functions with large number of variables. This is done by performing the simplification process in the frequency domain rather than time domain. The proposed method can be implemented by using parallel processors. As a result, fast computing and simplification can be achieved.

H. M. El-Bakry is assistant professor with Dept. of Information Systems - Faculty of Computer Science and Information Systems – Mansoura University – Egypt. (phone: +2-050-2349340, fax: +2-050-2221442, e-mail: helbakry20@yahoo.com).

II. FAST TERM DETECTION BY USING CROSS CORRELATION IN THE FREQUENCY DOMAIN

Finding a group of ones in the input two dimensional matrix is a searching problem. Each position in the input matrix is tested for the presence or absence of group of ones. At each position in the input matrix, each sub-matrix is multiplied by a window of ones, which has the same size as the sub-matrix. When the final output is maximum, this means that the sub-matrix under test contains ones and vice versa. Thus, we may conclude that this searching problem is a cross correlation between the matrix under test and the window of ones.

Here, a fast algorithm for detecting groups of ones based on two dimensional cross correlations that take place between the tested matrix and the sliding window is described. Such window is represented by a group of ones. The convolution theorem in mathematical analysis says that a convolution of f with h is identical to the result of the following steps: let F and H be the results of the Fourier transformation of f and h in the frequency domain. Multiply F and H in the frequency domain point by point and then transform this product into spatial domain via the inverse Fourier transform [56]. As a result, these cross correlations can be represented by a product in the frequency domain. Thus, by using cross correlation in the

frequency domain a speed up in an order of magnitude can be achieved during the searching process [36-54].

In the detection phase, a sub-matrix X of size mxz (sliding window) is extracted from the tested large input matrix, which has a size PxT. Let W be the group of ones matrix which has dimensions of mxz. The output can be calculated as follows:

$$h = \sum_{j=1}^{m} \sum_{k=1}^{z} W(j,k)X(j,k) \quad (1)$$

Eq. 1 represents the output for a particular sub-matrix *X*. It can be computed for the whole matrix Ψ as follows:

$$h(u,v) = \sum_{j=-m/2}^{m/2} \sum_{k=-z/2}^{z/2} W(j,k)\Psi(u+j,v+k) \quad (2)$$

Eq.(2) represents a cross correlation operation. Given any two functions *f* and *g*, their cross correlation can be obtained by [56]:

$$g(x,y) \otimes f(x,y) = \left(\sum_{m=-\infty}^{\infty} \sum_{z=-\infty}^{\infty} g(m,z)f(x+m,y+z) \right) \quad (3)$$

Therefore, Eq.(2) can be written as follows [36-54]:

$$h = W \otimes \Psi \quad (4)$$

here h is the output when the sliding window is located at position *(u,v)* in the input matrix Ψ and $(u,v) \in [P-m+1, T-n+1]$.

Now, the above cross correlation can be expressed in terms of the Fourier Transform:

$$W \otimes \Psi = F^{-1}\left(F(\Psi) \bullet F^{*}(W)\right) \quad (5)$$

(*) means the conjugate of the *FFT* for the group of ones matrix. Hence, by evaluating this cross correlation, a speed up ratio can be obtained comparable to conventional cross correlation.

The complexity of cross correlation in the frequency domain can be analyzed as follows:

1. For a tested matrix of *NxN* elements, the *2D-FFT* requires a number equal to $N^2 log_2 N^2$ of complex computation steps. The same number of complex computation steps required for computing the *2D-FFT* for the group of ones matrix can be done off line.

2. The inverse *2D-FFT* is computed. So, *(1+1)* forward transforms have to be computed. Therefore, for an matrix under test, the total number of the *2D-FFT* to compute is $2N^2 log_2 N^2$.

3. The input matrix and the group of ones matrix should be multiplied in the frequency domain. Therefore, a number of complex computation steps equal to qN^2 should be added.

4. The number of computation steps required by the fast cross correlatioon is complex and must be converted into a real version. It is known that the two dimensional Fast Fourier Transform requires $(N^2/2)log_2 N^2$ complex multiplications and $N^2 log_2 N^2$ complex additions [59]. Every complex multiplication is realized by six real floating point operations and every complex addition is implemented by two real floating point operations. So, the total number of computation steps required to obtain the *2D-FFT* of an *NxN* matrix is:

$$\rho = 6((N^2/2)log_2 N^2) + 2(N^2 log_2 N^2) \quad (6)$$

which may be simplified to:

$$\rho = 5N^2 log_2 N^2 \quad (7)$$

Performing complex dot product in the frequency domain also requires $6qN^2$ real operations.

5. In order to perform cross correlation in the frequency domain, the group of ones matrix must have the same size as the input matrix. Assume that the input object has a size of (nxn) dimensions. So, the search process will be done over sub-matrixes of (nxn) dimensions and the group of ones matrix will have the same size. Therefore, a number of zeros = (N^2-n^2) must be added to the group of ones matrix. This requires a total real number of computation steps = $q(N^2-n^2)$ for all neurons. Moreover, after computing the *2D-FFT* for the group of ones matrix, the conjugate of this matrix must be obtained. So, a real number of computation steps $=qN^2$ should be added in order to obtain the conjugate of the group of ones matrix for all neurons. Also, a number of real computation steps equal to *N* is required to create butterflies complex numbers $(e^{-jk(2\Pi n/N)})$, where $0<K<L$. These *(N/2)* complex numbers are multiplied by the elements of the input matrix or by previous complex numbers during the computation of the *2D-FFT*. To create a complex number requires two real floating point operations. So, the total number of computation steps required for the fast cross correlation becomes:

$$\sigma = (10N^2 log_2 N^2) + 6N^2 + (N^2-n^2) + N^2 + N \quad (8)$$

which can be reformulated as:

$$\sigma = (10N^2 log_2 N^2) + (8N^2-n^2) + N \quad (9)$$

6. Using a sliding window of size nxn for the same matrix of *NxN* elements, $(2n^2-1)(N-n+1)^2$ computation steps are required when using traditional cross correlation for the searching process. The theoretical speed up factor η can be evaluated as follows:

$$\eta = \frac{(2n^2-1)(N-n+1)^2}{(10N^2 log_2 N^2) + (8N^2-n^2) + N} \quad (10)$$

The theoretical speed up ratio Eq. 10 with different sizes of the input matrix and different in size group of ones matrixes is listed in Table 1. Practical speed up ratio for manipulating matrixes of different sizes and different in size group of ones matrixes is listed in Table 2 using 2.7 GHz processor and *MATLAB ver 5.3*. An interesting property with FNNs is that the number of computation steps does not depend on either the

TABLE I
THE THEORETICAL SPEED UP RATIO FOR KARNOUGH MAPS WITH DIFFERENT SIZES

Matrix size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	3.73	5.13	6.45
200x200	4.07	6.02	8.18
300x300	4.06	6.13	8.51
400x400	4.01	6.11	8.56
500x500	3.95	6.05	8.53
600x600	3.90	5.98	8.47
700x700	3.84	5.92	8.39
800x800	3.80	5.86	8.32
900x900	3.75	5.80	8.25
1000x1000	3.71	5.74	8.19
1100x1100	3.67	5.69	8.12
1200x1200	3.64	5.65	8.06
1300x1300	3.61	5.60	8.01
1400x1400	3.59	5.56	7.95
1500x1500	3.56	5.53	7.90
1600x1600	3.54	5.49	7.86
1700x1700	3.51	5.46	7.81
1800x1800	3.49	5.43	7.77
1900x1900	3.47	5.40	7.73
2000x2000	3.45	5.37	7.69

TABLE II
PRACTICAL SPEED UP RATIO FOR KARNOUGH MAPS WITH DIFFERENT SIZES USING MATLAB VER 5.3

Matrix size	Speed up ratio (n=20)	Speed up ratio (n=25)	Speed up ratio (n=30)
100x100	5.34	8.08	11.97
200x200	4.02	7.13	10.54
300x300	3.49	6.59	9.99
400x400	2.89	6.18	9.31
500x500	2.67	5.95	9.96
600x600	2.49	5.82	9.38
700x700	2.38	5.71	8.99
800x800	2.29	5.59	8.78
900x900	2.33	5.78	8.98
1000x1000	2.19	5.63	8.76
1100x1100	2.25	5.60	8.64
1200x1200	2.22	5.57	8.56
1300x1300	2.18	5.54	8.50
1400x1400	2.15	5.50	8.45
1500x1500	2.11	5.46	8.40
1600x1600	2.08	5.42	8.36
1700x1700	2.05	5.39	8.32
1800x1800	2.02	5.36	8.28
1900x1900	1.99	5.32	8.24
2000x2000	1.96	5.29	8.21

size of the input sub-matrix or the size of the group of ones matrix (n). The effect of (n) on the number of computation steps is very small and can be ignored. This is in contrast to conventional cross correlation in which the number of computation steps is increased with the size of both the input sub-matrix and the group of ones matrix (n).

III. Implementation of Simplified Functions by using MNNs

Here, a powerful solution for realization of complex functions is given. The simplified functions are implemented by using a new desigen for neural networks. Neural networks are used because they are fault tolerance. Therefore, they can recognize signals even with noise or distortion. This is very useful for logic functions used in data and computer communications. The implemented functions are realized with minimum amount of components. MNNs present a new trend in neural network architecture design. Motivated by the highly-modular biological network, artificial neural net designers aim to build architectures which are more scalable and less subjected to interference than the traditional non-modular neural nets [57]. There are now a wide variety of MNN designs for classification. Non-modular classifiers tend to introduce high internal interference because of the strong coupling among their hidden layer weights [58]. As a result of this, slow learning or over fitting can be done during the learning process. Sometime, the network could not be learned for complex tasks. Such tasks tend to introduce a wide range of overlap which, in turn, causes a wide range of deviations from efficient learning in the different regions of input space [60]. Usually there are regions in the class feature space which show high overlap due to the resemblance of two or more input patterns (classes). At the same time, there are other regions which show little or even no overlap, due to the uniqueness of the classes therein. High coupling among hidden nodes will then, result in over and under learning at different regions [64]. Enlarging the network, increasing the number and quality of training samples, and techniques for avoiding local minina, will not stretch the learning capabilities of the NN classifier beyond a certain limit as long as hidden nodes are tightly coupled, and hence cross talking during learning [58]. A MNN classifier attempts to reduce the effect of these problems via a divide and conquer approach. It, generally, decomposes the large size / high complexity task into several sub-tasks, each one is handled by a simple, fast, and efficient module. Then, sub-solutions are integrated via a multi-module decision-making strategy. Hence, MNN classifiers, generally, proved to be more efficient than non-modular alternatives [62]. However, MNNs can not offer a real alternative to non-modular networks unless the MNNs designer balances the simplicity of subtasks and the efficiency of the multi module decision-making strategy. In other words, the task decomposition algorithm should produce sub tasks as they can be, but meanwhile modules have to be able to give the multi module decision making strategy enough information to take accurate global decision [60,61]. In previous papers [52-54], it has been shown that this model can be applied to realize non-binary data. In this paper, it is proven that MNNs can solve some problems with a little amount of requirements than non-MNNs. In section 2, XOR function, and 16 logic functions on one bit level are simply implemented using MNN. Comparisons with conventional MNN are given. In section 3, another strategy for the design of MNNS is presented and applied to realize, and 2-bit digital multiplier.

IV. Complexity Reduction Using Modular Neural Networks

In the following subsections, we investigate the usage of MNNs in some binary problems. Here, all MNNs are feedforward type, and learned by using backpropagation algorithm. In comparison with non-MNNs, we take into account the number of neurons and weights in both models as well as the number of computations during the test phase.

A) *A simple implementation of XOR problem*

There are two topologies to realize XOR function whose truth Table is shown in Table 3 using neural nets. The first uses fully connected neural nets with three neurons, two of which are in the hidden layer, and the other is in the output layer. There is no direct connections between the input and output layer as shown in Fig.1. In this case, the neural net is trained to classify all of these four patterns at the same time.

Table III
Truth Table Of XOR Function.

x	y	O/P
0	0	0
0	1	1
1	0	1
1	1	0

The second approach was presented by Minsky and Papert which was realized using two neurons as shown in Fig. 2. The first representing logic AND and the other logic OR. The value of +1.5 for the threshold of the hidden neuron insures that it will be turned on only when both input units are on. The value of +0.5 for the output neuron insures that it will turn on only when it receives a net positive input greater than +0.5. The weight of -2 from the hidden neuron to the output one insures that the output neuron will not come on when both input neurons are on [63]. Using MNNs, we may consider the problem of classifying these four patterns as two individual problems. This can be done at two steps:

1- We deal with each bit alone.
2- Consider the second bit Y, Divide the four patterns into two groups.

The first group consists of the first two patterns which realize a buffer, while the second group which contains the other two patterns represents an inverter as shown in Table 4. The first bit (X) may be used to select the function.

TABLE IV
RESULTS OF DIVIDING XOR PATTERNS

X	Y	O/P	New Function
0 0	0 1	0 1	Buffer (Y)
1 1	0 1	1 0	Inverter ($\overline{Y}$)

So, we may use two neural nets, one to realize the buffer, and the other to represent the inverter. Each one of them may be implemented by using only one neuron. When realizing these two neurons, we implement the weights, and perform only one summing operation. The first input X acts as a detector to select the proper weights as shown in Fig.3. In a special case, for XOR function, there is no need to the buffer and the neural net may be represented by using only one weight corresponding to the inverter as shown in Fig.4. As a result of using cooperative modular neural nets, XOR function is realized by using only one neuron. A comparison between the new model and the two previous approaches is given in Table 5. It is clear that the number of computations and the hardware requirements for the new model is less than that of the other models.

TABLE V
A COMPARISON BETWEEN DIFFERENT MODELS USED TO IMPLEMENT XOR FUNCTION

Type of Comparison	First model (three neurons)	Second model (two neurons)	New model (one neuron)
No. of computations	O(15)	O(12)	O(3)
Hardware requirements	3 neurons, 9 weights	2 neurons, 7 weights	1 neuron, 2 weights, 2 switches, 1 inverter

B) Implementation of logic Function using MNN

Realization of logic functions in one bit level (X,Y) generates 16 functions which are (AND, OR, NAND, NOR, XOR, XNOR, $\overline{X}$, $\overline{Y}$, X, Y, 0, 1, $\overline{X}$ Y, X $\overline{Y}$, $\overline{X}$ +Y, X+ $\overline{Y}$). So, in order to control the selection for each one of these functions, we must have another 4 bits at the input, thereby the total input is 6 bits as shown in Table 6.

TABLE VI
TRUTH TABLE OF LOGIC FUNCTION (ONE BIT LEVEL) WITH THEIR CONTROL SELECTION

Function	C1	C2	C3	C4	X	Y	O/p
AND	0	0	0	0	0	0	0
	0	0	0	0	0	1	0
	0	0	0	0	1	0	0
	0	0	0	0	1	1	1
-	-	-	-	-	-	-	-
X+ $\overline{Y}$	1	1	1	1	0	0	1
	1	1	1	1	0	1	0
	1	1	1	1	1	0	1
	1	1	1	1	1	1	1

Non-MNNs can classify these 64 patterns using a network of three layers. The hidden layer contains 8 neurons, while the output needs only one neuron and a total number of 65 weights are required. These patterns can be divided into two groups. Each group has an input of 5 bits, while the MSB is 0 with the first group and 1 with the second. The first group requires 4 neurons and 29 weights in the hidden layer, while the second needs 3 neurons and 22 weights. As a result of this, we may implement only 4 summing operations in the hidden layer (in spite of 8 neurons in case of non-MNNs) where as the MSB is used to select which group of weights must be connected to the neurons in the hidden layer. A similar procedure is done between hidden and output layer. Fig. 5 shows the structure of the first neuron in the hidden layer. A comparison between MNN and non-MNNs used to implement logic functions is shown in Table 7.

TABLE VII
A COMPARISON BETWEEN MNNS AND NON MNNS USED TO IMPLEMENT 16 LOGIC FUNCTIONS

Type of Comparison	Realization using non MNNs	Realization using MNNs
No. of computations	O(121)	O(54)
Hardware requirements	9 neurons, 65 weights	5 neurons, 51 weights, 10 switches, 1 inverter

V. IMPLEMENTATION OF 2-BITS DIGITAL MULTIPLIER USING MNNS

In the previous section, to simplify the problem, we make division in input, here is an example for division in output. According to the truth table shown in Table 8, instead of treating the problem as mapping 4 bits in input to 4 bits in output, we may deal with each bit in output alone. Non MNNs can realize the 2-bits multiplier with a network of three layers and a total number of 31 weights. The hidden layer contains 3 neurons, while the output one has 4 neurons. Using MNN we may simplify the problem as:

$$W = CA \tag{11}$$

$$X = AD \otimes BC = AD(\overline{B} + \overline{C}) + BC(\overline{A} + \overline{D}) = (AD + BC)(\overline{A} + \overline{B} + \overline{C} + \overline{D}) \tag{12}$$

$$Y = BD(\overline{A} + \overline{C}) = BD(\overline{A} + \overline{B} + \overline{C} + \overline{D}) \tag{13}$$

$$Z = ABCD \tag{14}$$

Equations 1, 2, 3 can be implemented using only one neuron. The third term in Equation 3 can be implemented using the output from Bit Z with a negative (inhibitory) weight. This eliminates the need to use two neurons to represent $\overline{A}$ and $\overline{D}$. Equation 2 resembles an XOR, but we must first obtain AD and BC. AD can be implemented using only one neuron. Another neuron is used to realize BC and at the same time oring (AD, BC) as well as anding the result with ($\overline{ABCD}$) as shown in Fig. 6 . A comparison between MNN and non-MNNs used to implement 2bits digital multiplier is listed in Table 9.

VI. Conclusion

A fast simplification method has been presented. Karnough map with large number of variables has been realized. The presented idea depends on fast detection of group of ones in the visualized map. This has been done by performing cross correlation in the frequency domain rather than time one. It is proved mathematically and practically that the number of computation steps required for the presented method is less than that needed by conventional cross correlation. Simulation results using MATLAB confirm the theoretical computations. Furthermore, it can be implemented by using parallel processors. In addition, a new model for realizing complex function has been presented. Such model realies on MNNs neural nets for classifying patterns that appeared expensive to be solved by using conventional models of neural nets. This approach has been introduced to realize different types of logic functions. Moreover, it can be applied to manipulate non-binary data. Compared to non MNNS, realization of problems using MNNs resulted in reduction of the number of computations, neurons and weights.

Table 8: Truth table of 2-bit digital multiplier.

Input Patterns				Output Patterns			
D	C	B	A	Z	Y	X	W
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Table 9: A comparison between MNN and non-MNNs used to implement 2-bits digital multiplier.

Type of Comparison	Realization using non MNNs	Realization using MNNs
No. of computations	O(55)	O(35)
Hardware requirements	7 neurons, 31 weights	5 neurons, 20 weights

References

[1] Ledion Bitincka, George E. Antoniou, “PDA-based Boolean Function Simplification: A Useful Educational Tool,” Informatika, 2004, Vol. 15, no. 3, pp. 329-336.

[2] Ledion Bitincka, George E. Antoniou, “Pocket-PC Boolean Function Simplification,” Journal of Electrical Engineering, vol. 56, no. 7-8, 2005, pp. 209–212.

[3] TOMASZEWSKI, S. P.—ILGAZ, I. U.—ANTONIOU, G. E. :WWW-Based Boolean Function Simplification, International Journal of Mathematics and Computer Science 13 No. 4 (2003), 577–583.

[4] WAKERLY, J. F. : Digital Design, Prentice-Hall, New York, 2000.

[5] NELSON, V. P.—NAGLE, H. T.—CARROLL, B. D.—IRWIN, D. : Digital Logic Circuit Analysis and Design, Prentice-Hall, New Jersey, 1995.

[6] KATZ, R. : Contemporary Logic Design, Benjamin/Cummings Publ, Redwood City, CA, 1994.

[7] BROWN, S.—VRANESIC, Z. : Fundamentals of Digital Logic with VHDL, McGraw-Hill, New York, 2003.

[8] HAYES, J. P. : Digital Logic Design, New York, 1993.

[9] KARNAUGH, M.: The Map Method for Synthesis of Com-binatorial Logic Circuits, Trans. AIEE, Communications and Electronics 72 (1953), 593-598.

[10] QUINE, W. V. : The Problem of Simplifying Truth Tables, Am. Math. Monthly 59 No. 8 (1952), 521–531.

[11] McCLUSKEY, E. J. : Minimization of Boolean functions, Bell System Tech. Journal 35 No. 5 (1956), 1417–1444.

[12] GAJSKI, D. D. : Principles of Digital Design, Prentice-Hall, 1997.

[13] CHIRLIAN, P. M. : Digital Circuits with Microprocessor Applications, Matrix Publishers, Oregon, 1982.

[14] HILL, F. J,—PETERSON, G. R. : Computer Aided Logical Design with Emphasis on VLSI, Wiley, New York, 1993.

[15] Alan B. Marcovitz, "Introduction to Logic and Computer Design "Hardcover, 2007.

[16] Alan B. Marcovitz, "Introduction to Logic Design," (2nd Economy Edition), Paperback, 2005.

[17] Amy E. Arntson, "Digital Design Basics," Paperback, 2005.

[18] Frank Vahid, " Digital Design," Hardcover, 2006.

[19] Janaye M. Houghton and Robert S. Houghton, "Circuit Sense for Elementary Teachers and Students: Understanding and Building Simple Logic Circuits," Paperback, 1994.

[20] John F. Wakerly, "Digital Design: Principles and Practices Package (4th Edition), Hardcover, 2005.

[21] Mano, Charles Kime, "Logic and Computer Design Fundamentals" (Third Edition), Hardcover, 2003.

[22] Morris M. Mano, Michael D. Ciletti, "Digital Design (4th Edition), Hardcover, 2006.

[23] Morris M. Mano, "Digital Design," Hardcover, 1984.

[24] Morris M. Mano, "Computer System Architecture (3rd Edition)", Hardcover, 1992.

[25] Nripendra N. Biswas, "Computer aided minimization procedure for boolean functions," Proceedings of the 21st conference on Design automation," Albuquerque, New Mexico, United States, Pages: 699 – 702, 1984.

[26] Randal E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers, C-35-8, pp. 677-691, August, 1986.

[27] Robert Dueck, "Digital Design with CPLD Applications and VHDL(Digital Design: Principles and Practices Package (2nd Edition), Hardcover, 2004.
[28] Thomas L. Floyd ,"Electronics Fundamentals: Circuits, Devices and Applications" (7th Edition), Hardcover, 2006.
[29] Thomas L. Floyd, Digital Fundamentals, Hardcover, 1994.
[30] Victor P. Nelson, H. Troy Nagle, Bill D. Carroll, and David Irwin, "Digital Logic Circuit Analysis and Design", Paperback, 1995
[31] William Kleitz, "Digital and Microprocessor Fundamentals: Theory and Application (4th Edition), Hardcover, 2002.
[32] Yves Crama and Peter L. Hammer, "Boolean Functions Theory, Algorithms and Applications", January 22, 2006.
[33]http://www.asic-world.com/digital/kmaps.html, "Simplification of Boolean Functions", 2006.
[34] Hazem M. El-Bakry, and Ahmed Atwan, "Simplification and Implementation of Boolean Functions," International Journal of Universal Computer Sciences, issue 1, vol. 1, 2010, pp. 19-33.
[35] Hazem M. El-bakry, Ahmed Atwan, and Nikos Mastorakis "A New Technique for Realization of Boolean Functions, " Proc. of Recent Advances in Artificial Intelligence, Koweledge Engineering and Databases, Cambridge, UK, February 20-22, 2010, pp. 260-270.
[36] Hazem M. El-bakry, and Nikos Mastorakis "A Fast Computerized Method For Automatic Simplification of Boolean Functions," Proc. of 9th WSEAS International Conference on SYSTEMS THEORY AND SCIENTIFIC COMPUTATION (ISTASC '09), Moscow, Russia, August 26-28, 2009, pp. 99-107.
[37] Hazem M. El-bakry, "Fast Karnough Map for Simplification of Complex Boolean Functions," Proc. of 10th WSEAS International Conference on Applied Computer Science (ACS'10), Japan, October 4-6, 2010, pp. 478-483.
[38] Hazem M. El-Bakry, "An Efficient Algorithm for Pattern Detection using Combined Classifiers and Data Fusion," Information Fusion Journal, vol. 11, 2010, pp. 133-148..
[39] Hazem M. El-Bakry, "A Novel High Speed Neural Model for Fast Pattern Recognition," Soft Computing Journal, vol. 14, no. 6, 2010, pp. 647-666.
[40] Hazem M. El-Bakry, "Fast Virus Detection by using High Speed Time Delay Neural Networks," Journal of Computer Virology, vol.6, no.2, 2010, pp.115-122.
[41] Hazem M. El-Bakry, "New Fast Principal Component Analysis For Real-Time Face Detection," MG&V Journal, vol. 18, no. 4, 2009, pp. 405-426.
[42] Hazem M. El-Bakry, "A New Neural Design for Faster Pattern Detection Using Cross Correlation and Matrix Decomposition," Neural World journal, 2009, vol. 19, no. 2, pp. 131-164.
[43] Hazem M. El-Bakry and M. Hamada, "A New Implementation for High Speed Neural Networks in Frequency Space," Lecture Notes in Computer Science, Springer, KES 2008, Part I, LNAI 5177, pp. 33-40.
[44] Hazem M. El-Bakry and Mohamed Hamada, " New Fast Decision Tree Classifier for Identifying Protein Coding Regions," Lecture Notes in Computer Science, Springer, ISICA 2008, LNCS 5370, 2008, pp. 489-500.
[45] Hazem M. El-Bakry, and Nikos Mastorakis "New Fast Normalized Neural Networks for Pattern Detection," Image and Vision Computing Journal, vol. 25, issue 11, 2007, pp. 1767-1784.
[46] Hazem M. El-Bakry and Nikos Mastorakis, "Fast Code Detection Using High Speed Time Delay Neural Networks," Lecture Notes in Computer Science, Springer, vol. 4493, Part III, May 2007, pp. 764-773.
[47] Hazem M. El-Bakry, "New Fast Time Delay Neural Networks Using Cross Correlation Performed in the Frequency Domain," Neurocomputing Journal, vol. 69, October 2006, pp. 2360-2363.
[48] Hazem M. El-Bakry, "Face detection using fast neural networks and image decomposition," Neurocomputing Journal, vol. 48, 2002, pp. 1039-1046.
[49] Hazem M. El-Bakry, "Human Iris Detection Using Fast Cooperative Modular Neural Nets and Image Decomposition," Machine Graphics & Vision Journal (MG&V), vol. 11, no. 4, 2002, pp. 498-512.
[50] Hazem M. El-Bakry, "Automatic Human Face Recognition Using Modular Neural Networks," Machine Graphics & Vision Journal (MG&V), vol. 10, no. 1, 2001, pp. 47-73.
[51] Hazem M. El-Bakry, "New Faster Normalized Neural Networks for Sub-Matrix Detection using Cross Correlation in the Frequency Domain and Matrix Decomposition," Applied Soft Computing journal, vol. 8, issue 2, March 2008, pp. 1131-1149.
[52] Hazem M. El-Bakry, "Automatic Human Face Recognition Using Modular Neural Networks," Machine Graphics & Vision Journal (MG&V), vol. 10, no. 1, 2001, pp. 47-73.
[53] Hazem M. El-Bakry, "Human Iris Detection Using Fast Cooperative Modular Neural Nets and Image Decomposition," Machine Graphics & Vision Journal (MG&V), vol. 11, no. 4, 2002, pp. 498-512.
[54] Hazem M. El-Bakry "Fast Iris Detection for Personal Verification Using Modular Neural Networks," Lecture Notes in Computer Science, Springer, vol. 2206, October 2001, pp. 269-283.
[55] Hazem M. El-bakry, "Complexity Reduction Using Modular Neural Networks," Proc. of IEEE IJCNN'03, Portland, Oregon, pp. 2202-2207, July, 20-24, 2003.
[56] R. Klette, and Zamperon, "Handbook of image processing operators, " John Wiley & Sonsltd, 1996.
[57] J. Murre, Learning and Categorization in Modular Neural Networks, Harvester Wheatcheaf. 1992.
[58] R. Jacobs, M. Jordan, A. Barto, Task Decomposition Through Competition in a Modular Connectionist Architecture: The what and where vision tasks, Neural Computation 3, pp. 79-87, 1991.
[59] J. W. Cooley, and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comput. 19, 297–301, 1965.
[60] G. Auda, and M. Kamel, CMNN: Cooperative Modular Neural Networks for Pattern Recognition, Pattern Recognition Letters, Vol. 18, pp. 1391-1398, 1997.

[61] E. Alpaydin, , Multiple Networks for Function Learning, Int. Conf. on Neural Networks, Vol.1 CA, USA, pp. 9-14, 1993.
[62] A. Waibel, Modular Construction of Time Delay Neural Networks for Speach Recognition, Neural Computing 1, pp.39-46.
[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representation by error backpropagation, Parallel distributed Processing: Explorations in the Microstructues of Cognition, Vol. 1, Cambridge, MA:MIT Press, pp. 318-362, 1986.
[64] K. Joe, Y. Mori, S. Miyake, Construction of a large scale neural network: Simulation of handwritten Japanese Character Recognition, on NCUBE Concurrency 2 (2), pp. 79-107.

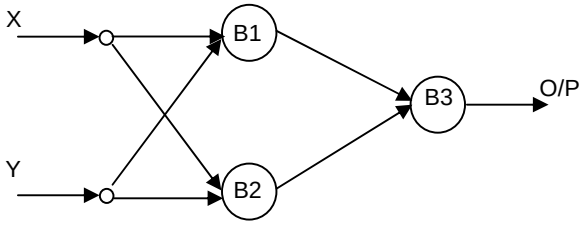

Fig. 1 Realization of XOR function using three neurons

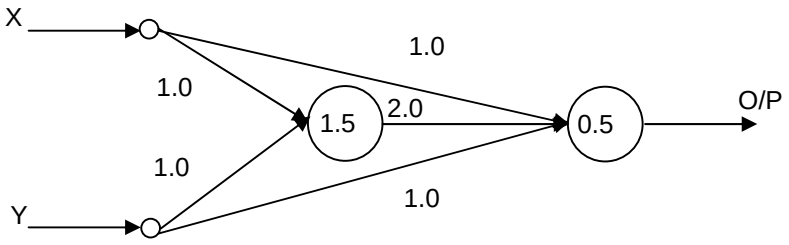

Fig. 2 Realization of XOR function using two neurons.

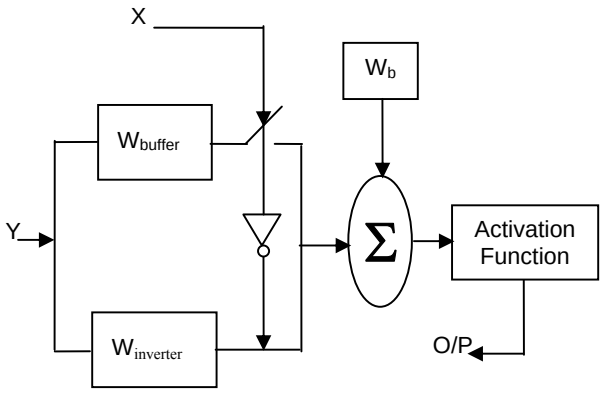

Fig. 3 Realization of XOR function using modular neural nets

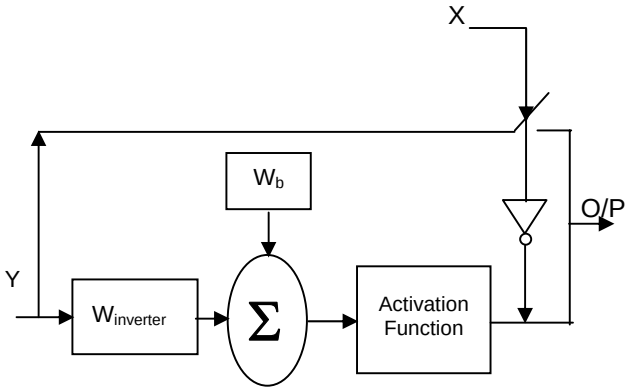

Fig. 4 Implementation of xor function using only one neuron

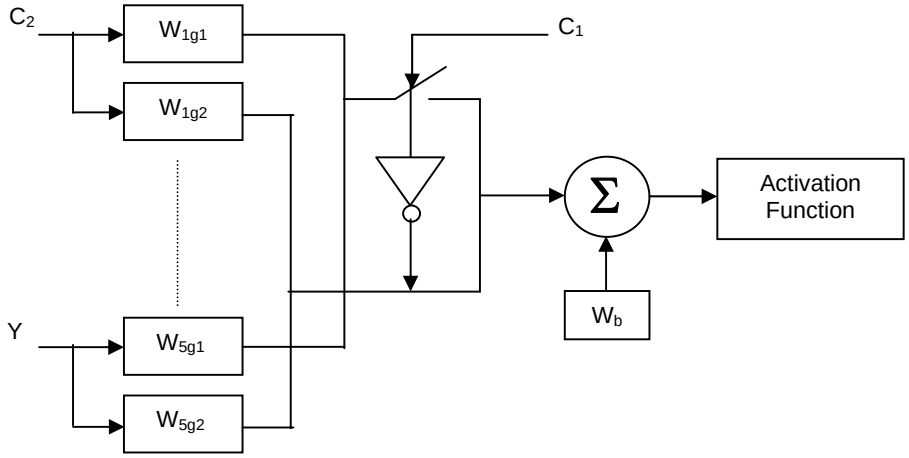

Fig. 5 Realization of logic functions using MNNs (the first neuron in the hidden layer).

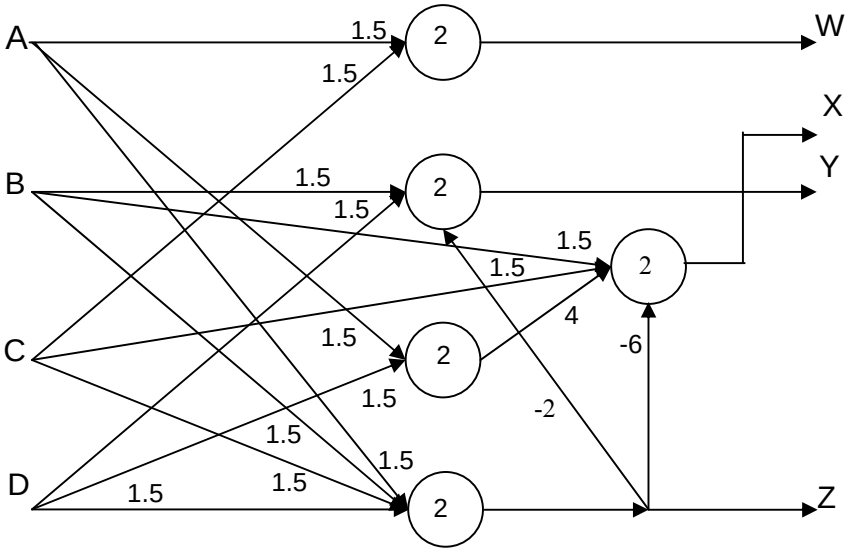

Fig. 6 Realization of 2-bits digital multiplier using MNNs.