Mansoura Engineering Journal

Volume 18 | Issue 3 Article 4

4-20-2021

# Power Series Representation of Logical Functions and its Applications to Error Detection and Error Correction Codes.

Yehia Enab
*Electronics and Communication Engineering Department., Faculty of Engineering., El-Mansoura University.,Mansoura., Egypt.*

Fayez Zaki
*Associate Professor., Electronics and Communication Engineering Department., Faculty of Engineering., El-Mansoura University.,Mansoura., Egypt.*, fwzaki@mans.edu.eg

Mansoura Engineering Journal (MEJ), Vol. 18, No, 3 Sept. 1993 E. 1

# POWER SERIES REPRESENTATION OP LOGICAL FUNCTIONS AND ITS APPLICATIONS TO ERROR DETECTION AND ERROR CORRECTION CODES

Yebia M. I. Enab and Fayez W. Zaki

Faculty of Engineering, Mansoura University, Egypt.

## تمثيل الدوال المنطقية باستخدام مفكوك متواليات القوى و تطبيقاتها فى بناء دوائر إكتشاف و تصحيح خطأ التشفير

خلاصة :

معظم نظم تخزين المعلومات الرقمية و كذلك الإتصالات الرقمية تزود بدوائر لتحديد الخطأ فى المعلومات المشفرة و كيفية تصحيح هذا الخطأ . و يتم بناء هذه الدوائر باستخدام المسجلات المتتابعة ذات التغذية الخلفية . و بالرغم من أن هذه الدوائر قليلة التكلفة إلا أنها بطيئة جدا و يمكن الحصول على دوائر أسرع لمثل هذا الغرض باستخدام الدوائر المنطقية المتوازية ذات المسجلات المتتابعة . و لكن لسوء الحظ فإن مثل هذه الدوائر تكون معقدة جدا و كثيرة التكلفة .

هذا البحث يقدم طريقة مستحدثة لتحويل الدوائر المنطقية إلى صورة مفكوك متواليات القوى و هذه الطريقة تعمل على تسهيل بناء الدوائر المنطقية باستخدام الدوائر المتكاملة الخطية و التى لا تحتاج إلى إزاحة المعلومات بالتتابع كما فى الدوائر السابقة و إنما يتم الحصول على المعلومات المطلوبة بمجرد دخول المعلومات المناظرة . و ينتهى البحث بتصميم أحد دوائر أنظمة تصحيح الخطأ فى المعلومات المشفرة باستخدام الطريقة المستحدثة .

KEYWORDS:Neural network,switching circuits, computer interfacing, error detection and error correction, encoders and decoders.

ABSTRACT

The need for high speed digital storage/retrieval and digital communication systems with error detection and error correction capabilities raises a significant problem to the implementation of error control encoding and decoding. Traditional implementation has been accomplished through the use of feedback shift register networks. These networks, although being convenient to implement, represent a slow process in the overall system since they involve the shifting of code words to generate the error control bits/syndrome. Parallel implementation provides a fast solution, however, it poses a significant complexity to the encoder/decoder implementation.

This paper introduces a new implementation for a fast encoder/decoder using analog neural networks. With neural nets as the basic building blocks, the error control bits/syndrome are asynchronously generated using Op Amps and resistor elements. The problem of fan-in and fan-out inherent in digital implementation is also avoided.

## I-INTRODUCTION

Neural network technique has received considerable attention in the last few years. It is found to be suitable for a number of applications such as automatic control [1], constrained optimisation [2], pattern recognition and adaptive filtering [3], and etc.

Among the various neural network techniques, two groups are widely accepted in most applications. The first one is the Hopfield model [2]. These are single layer networks with symmetric interneuron connections. Such neurons are implemented with nonlinear amplifiers (hardlimiters). Hopfield showed that the computation power of this model is strong when applied to certain combinational and optimal problems. By properly designing the connection weights (synapses), the model can be used to solve the coding problem. In this case, the neuron looks for a code word that is nearest to

the received word subject to the constraint of the code structure.

Another neural network model reported in the literatures is the multilayer feedforward model. A typical example of this model is Widrow's ADALINEs [3]. They have been applied to pattern recognitions and adaptive filtering with impressive results.

Error detection and error correction coding is a signal processing operation which is used for reliable transmission of digital information over noisy channels. The technique rely upon the systematic addition of redundant bits to the transmitted code words so as to facilitate two basic objectives at the receiver: error detection and error correction [4,5]. This is performed as follows: the channel encoder accepts message bits and adds redundancy bits according to a prescribed rule, thereby producing encoded data at a higher bit rate. The channel decoder exploits the redundancy to decide which message bit was actually transmitted. The addition of redundancy in the coded messages implies the need for increased transmission bandwidth. Moreover, the use of coding adds complexity to the system, especially for the implementation of decoding operations in the receiver [6]. Thus the design trade-offs in the use of error control coding to achieve acceptable error performance must include considerations of bandwidth and system complexity.

Conventional implementation of error detection/correction circuits has been accomplished using feedback shift registers. These circuits although being convenient to implement, represent a slow process in the overall system. Parallel implementation provides a fast solution, however, it requires a significant amount of hardware complexity. In this paper, a new approach is introduced to simplify the implementation of asynchronous error detection/correction circuits using VLSI analog neural networks. The motivation behind analog VLSI is two fold: 1)VLSI is now maturing with emphasis towards submicron structures and sophisticated applications combining digital as well as analog circuits on a single chip, 2) massive applications of analog VLSI provides means for the hardware implementation of adaptive systems based on neural networks. Analog MOS circuits modules, such as integrators, summers, and multipliers can be configured in a neural network architecture to build feedback/feedforward neural networks and/or the equivalent of adaptive signal processors.

## II-POWER SERIES REPRESENTATION OF LOGIC FUNCTIONS

The analysis for this new concept may be introduced as follows: assuming that the binary inputs to a logical network are $\{x_i,\ i=1,2,\ldots N\}$ and the corresponding binary outputs are $\{y_j,\ j=1,2,\ldots M\}$. Each output $y_j$ may be expressed as

$$y_j = f(x_1,\ x_2,\ \ldots\ldots\ x_N) \qquad (1)$$

where f(.) is a mathematical function to be chosen as a power series expansion. Neglecting the subscript j for simplicity, Eq.(1) is expressed as:

$$y = \sum_{i_1=0}^{L_1} \sum_{i_2=0}^{L_2} \cdots \sum_{i_N=0}^{L_N} w_{i_1 i_2 i_3 \ldots i_N} x_1^{i_1} x_2^{i_2} \ldots\ldots x_N^{i_N} \qquad (2)$$

where $\{w_{i_1 i_2 \ldots\ldots i_N}\}$ are the coefficients (weights) of the power series and $L_1,\ L_2,\ \ldots\ldots\ L_N$ are integers chosen according to the desired accuracy of the representation. In Eq.(2);

$$x_k^{i_k} = x_k \qquad \text{for} \quad i_k > 1 \qquad (3)$$

This is due to the fact that $x_k$ is either 1 or 0. By applying Eq.(3) into Eq.(2), then

$$y = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_N=0}^{1} w_{i_1 i_2 i_3 \ldots\ldots i_N} x_1 x_2 \ldots\ldots x_N \qquad (4)$$

From Eq.(4), it can easily be seen that the maximum number of the coefficients to be determined is equal to $2^N$. This is the same number of entries in the corresponding truth table as defined in the classical switching theory. For example, let the logic circuit has two inputs $x_1$, $x_2$ and a single output y. The power series representation for such a circuit is expressed as

$$y = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} x_1^{i_1} x_2^{i_2} w_{i_1 i_2}$$

$$= w_{oo} x_1^{o} x_2^{o} + w_{o1} x_1^{o} x_2^{1} + w_{1o} x_1^{1} x_2^{o} + w_{11} x_1^{1} x_2^{1}$$

$$= w_{oo} + w_{1o} x_1 + w_{o1} x_2 + w_{11} x_1 x_2 \qquad (5)$$

On the other hand, power series expansion for a logic network can easily be performed from its truth table as explained in the following section.

## III-LOGICAL FUNCTION REPRESENTATIONS

The three basic logical functions are AND, OR, and NOT. The corresponding truth tables are:

AND Truth Table

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR Truth Table

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

NOT Truth Table

| x | y |
|---|---|
| 0 | 1 |
| 1 | 0 |

From the truth table of the NOT function, the output is given as;

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

similarly, from the truth table of the OR function, the output is;

$$y = \bar{x}_1 \text{ AND } x_2 \text{ OR } x_1 \text{ AND } \bar{x}_2 \text{ OR } x_1 \text{ AND } x_2$$

$$= (1 - x_1)x_2 + x_1(1 - x_2) + x_1 x_2$$

$$= x_1 + x_2 - x_1 x_2 \qquad (7)$$

and from the truth table of the AND function, the output is;

$$y = x_1 . x_2 \qquad (8)$$

where + , - , . are normal mathematical operations. Equations (6), (7), and (8) may be considered as curve fitting expressions for experimental data tabulated in the truth table.

In summary, the following algorithm is adopted to estimate the coefficients (synaptic weights) of the power series expansion for a given truth table:

Step 1

From the truth table express the logic function as a "sum of product" form;

Step 2

Replace each $\bar{x}$ by $(1 - x)$;

Step 3

Replace each ANDing operation ($x_1$ANDx$_2$) by mathematical product $x_1x_2$ and each ORing operation ($x_1$ORx$_2$) by $x_1 + x_2 - x_1x_2$;

Step 4

Simplify the resulting mathematical expression to the form given in Eq.(4);

Step 5

The coefficients of the power series resulted in step 3 are the weights required for circuit implementation.

Hardware implementations for the basic logical functions given in Eqs.(6), (7) and (8) using analog networks require analog multipliers and analog summers. Fig. 1, shows a two-input analog multiplier designed for logic inputs. This circuit is a modified configuration of the four-quadrant CMOS multiplier reported by Khachab and Ismail [12]. Using this multiplier circuit and the well known analog summer circuit, Eqs. (8), (6), and (7) are implemented as shown in Fig. (1), Fig. (2-a), and Fig. (2-b) respectively.

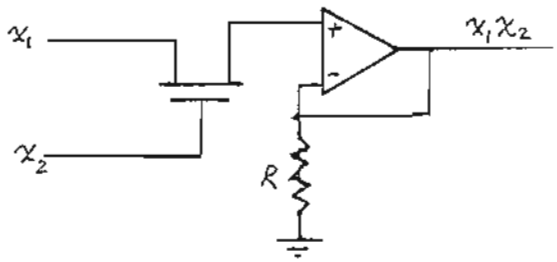


Fig. 1, A Two-Input Analog Multiplier (AND Circuit)

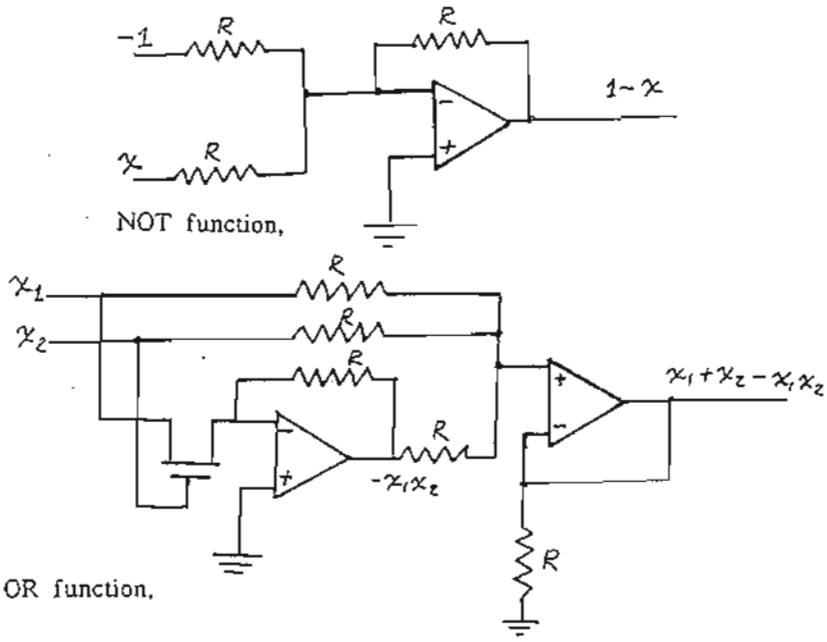


Fig. 2, Analog Implementation For The Basic Logic Functions.

Transient analysis for the circuits shown in Fig.(1) and Fig.(2-a) have been carried out using PSPICE simulation program. For the multiplier, $x_1$ was chosen as a 5 V. d.c. and $x_2$ as a square wave with a frequency of 50 KHz and 5 V. amplitude. This square wave was also used as the input for the inverter circuit shown in Fig. (2-a). Fig. (3) shows the simulation results using LM318 Op Amp and 2N6660 CMOS transistor. It can be seen that each circuit provides the function that was designed for.

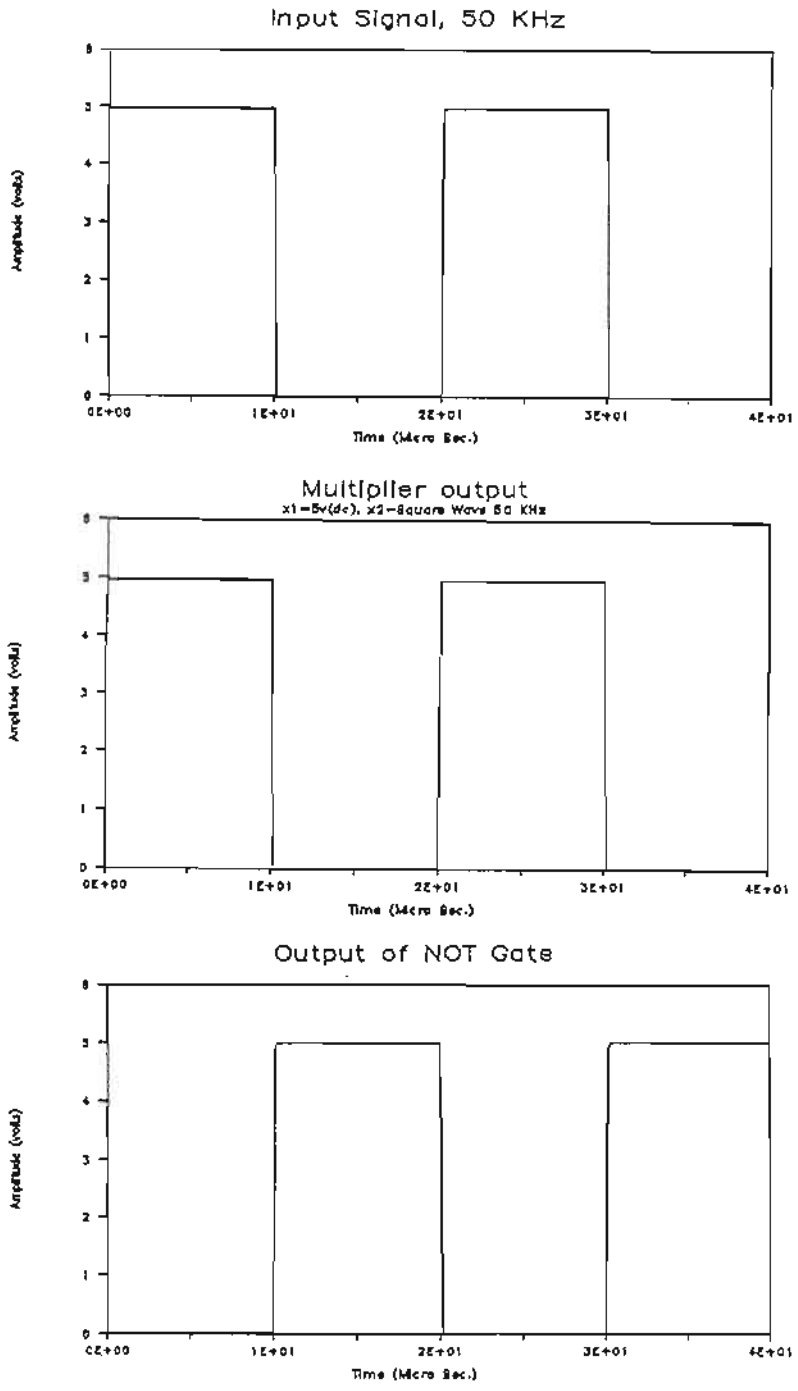


Fig. 3, a)Square Wave Input, b) Multiplier Output, and c)Inverter Output.

## IV-ADVANTAGES OF THE NEW CONCEPT

1-Logic function simplification using Karnaugh map is replaced by algebraic manipulation.

2-The use of Op Amps and resistor elements to implement the binary logic provides the possibility of high voltage and current levels. This

overcomes the difficulty of fan-in and fan-out associated with logic circuits.

3-The noise sensitivity can be eliminated by cascading the summer circuit with a comparator. The comparator provides an excellent buffering and supplies the communication channel by a stable binary signal.

## V-DISADVANTAGE OF THE NEW CONCEPT

1-For truth tables with large number of enteries, tedious mathematical manipulation is required.

2-The truth table should be complete in order to derive exact power series expansion. For partially known truth table, there is no guarantee that the derived expansion fits the unknown enteries of the table. This disadvantage is common in other methods for logical function representation.

## VI-LINEAR BLOCK CODES

According to [7] and as shown in Fig. 4, to generate an (n,k) block code, the channel encoder accepts information in successive k-bit blocks, for each block it adds (n - k) redundant bits that are algebraically related to the k message bits, thereby producing an overall encoded block of n bits, where n > k. The n bit block is called a code word, and n is called the block length of the code.

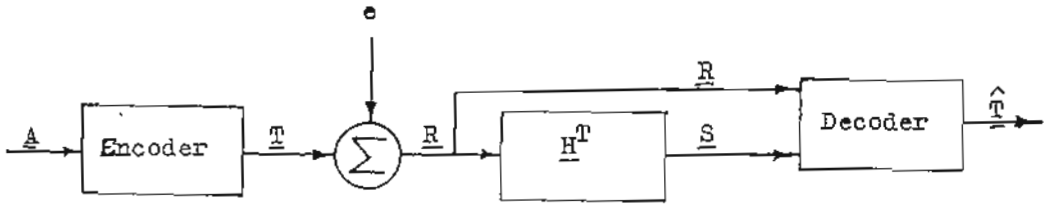


Fig. 4, Transmission of Digital Data with Block Codes

Consider (n,k) linear block code in which the first portion of k bits is always identical to the message sequence to be transmitted, Fig. 5. The (n-k) bits in the second portion are computed from the message bits in accordance with a prescribed encoding rule that determines the mathematical structure of the code [7,8]. These (n-k) bits are referred to as parity bits.

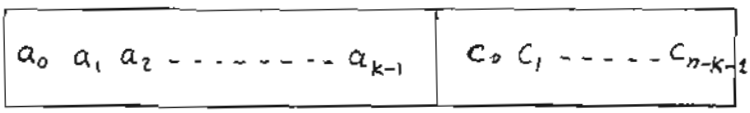


Fig. 5, Structure of Code word

Let $a_0, a_1, \ldots a_{k-1}$ constitute a block of k arbitrary message bits. Thus we have $2^k$ distinct message blocks. Let this sequence of message bits be applied to a linear block encoder, producing an n-bit code word whose elements are denoted by $t_0, t_1, \ldots t_{n-1}$. Let $c_0, c_1, \ldots c_{n-k-1}$ denote the (n-k) parity bits in the code word. Therefore,

$$t_i = \begin{cases} a_i & ; \quad i=0,1,2\ldots k-1 \\ c_{i-k} & ; \quad i=k,k+1,\ldots n-1 \end{cases} \qquad (9)$$

The (n-k) parity bits are linear sums of the k message bits, as given by the generalised equation

$$c_i = p_{io}a_o + p_{i1}a_1 + \ldots\ldots p_{i,k-1}a_{k-1} \; ; \; i=0,1,2,\ldots\ n-k-1 \qquad (10)$$

where the coefficients are defined as

$$p_{ij} = \begin{cases} 1 \; ; & \text{if } c_i \text{ depends on } a_i \\ 0 \; ; & \text{otherwise} \end{cases} \qquad (11)$$

These coefficients are chosen in such a way that the (n-k) equations represented in Eq.(10) are linearly independent, that is, no equation in the set can be expressed as a linear combination of the remaining ones.

The system of Eqs.(9) and (10) defines the mathematical structure of the (n,k) linear block code. The above equations may be rewritten in a compact form using matrix notation. To do so, we define the 1-by-k message row vector $\underline{A}$, the 1-by-(n-k) parity row vector $\underline{C}$ and the 1-by-n row code vector $\underline{T}$ as follows respectively:

$$\underline{A} = [\, a_o \;\; a_1 \;\ldots\ldots a_{k-1} \,] \qquad (12)$$

$$\underline{C} = [\, c_o \;\; c_1 \;\ldots\ldots c_{n-k-1} \,] \qquad (13)$$

$$\underline{T} = [\, t_o \;\; t_1 \;\ldots\ldots t_{n-1} \,] \qquad (14)$$

Note that, the use of row vectors is adopted for the sake of being consistent with the notation commonly used in the coding literatures [9]. This is different from that used in matrix algebra. Now, we may write the set of simultaneous equations in (10) as

$$\underline{C} = \underline{A}\,\underline{P} \qquad (15)$$

where $\underline{P}$ is the k-by-(n-k) coefficient matrix defined by

$$\underline{P} = \begin{bmatrix} p_{oo} & p_{1o}\cdots\cdots p_{n-k-1,o} \\ p_{o1} & p_{11}\cdots\cdots p_{n-k-1,1} \\ \multicolumn{2}{c}{\ldots\ldots\ldots\ldots\ldots\ldots} \\ p_{o,k-1} & p_{1,k-1}\cdots p_{n-k-1,k-1} \end{bmatrix} \qquad (16)$$

From the definitions given in Eq.(12) and (14), we see that $\underline{T}$ may be expressed as a partitioned row vector as:

$$\underline{T} = [\, \underline{A} \,|\, \underline{C} \,] \qquad (17)$$

Substituting Eq.(15) into Eq.(17), then

$$\underline{T} = [\, \underline{A} \,|\, \underline{A}\,\underline{P} \,] = \underline{A}\,[\, \underline{I}_k \,|\, \underline{P} \,] \qquad (18)$$

where $\underline{I}_k$ is the k-by-k identity matrix

$$\underline{I}_k = \begin{bmatrix} 1\ 0\ 0\ 0\ldots\ldots 0 \\ 0\ 1\ 0\ 0\ldots\ldots 0 \\ \ldots\ldots\ldots\ldots\ldots \\ 0\ 0\ 0\ 0\ldots\ldots 1 \end{bmatrix} \qquad (19)$$

Define the k-by-n generator matrix $\underline{G}$ as

$$\underline{G} = [\, \underline{I}_k \,\vdots\, \underline{P} \,] \tag{20}$$

Therefore, Eq.(18) is simplified to

$$\underline{T} = \underline{A}\ \underline{G} \tag{21}$$

Let us define an (n-k)-by-n matrix $\underline{H}$ as:

$$\underline{H} = [\, \underline{P}^T \,\vdots\, \underline{I}_{n-k} \,] \tag{22}$$

where $\underline{I}_{n-k}$ is an (n-k)-by-(n-k) identity matrix. Therefore,

$$\underline{H}\ \underline{G}^T = [\, \underline{P}^T \,\vdots\, \underline{I}_{n-k} \,] \begin{bmatrix} \underline{I}_k \\ \hline \underline{P}^T \end{bmatrix} = \underline{P}^T \oplus \underline{P}^T \tag{23}$$

Note that all additions are modulo-2 additions and all multiplications are AND operations. Hence

$$\underline{H}\ \underline{G}^T = \underline{G}\ \underline{H}^T = \underline{0} \tag{24}$$

Postmultiplying both sides of Eq.(21) by $\underline{H}^T$, then

$$\underline{T}\ \underline{H}^T = \underline{A}\ \underline{G}\ \underline{H}^T = \underline{0} \tag{25}$$

The matrix $\underline{H}$ is called the parity-check matrix of the code, and the Eqs.(25) are called parity-check equations.

The generator matrix $\underline{G}$ is used in the encoding operation at the transmitter. On the other hand, the parity-check matrix $\underline{H}$ is used in the decoding operation at the receiver. Let $\underline{R}$ denote the 1-by-n received vector that results from sending the code vector $\underline{T}$ over a noisy channel. That is;

$$\underline{R} = [r_1\ r_2\ \ldots\ldots\ r_n] \tag{26}$$

The vector $\underline{R}$ is expressed as the sum of the original code vector $\underline{T}$ and a vector $\underline{E}$ as [9]

$$\underline{R} = \underline{T} \oplus \underline{E} \tag{27}$$

The vector $\underline{E}$ is called the error vector or error pattern. The ith. element of $\underline{E}$ equals 0 if the corresponding element of $\underline{R}$ is the same as that of $\underline{T}$. On the other hand, the ith. element of $\underline{E}$ equals 1 if the corresponding element of $\underline{R}$ is different from that of $\underline{T}$, in which case an error is said to have occured in the ith. location. The receiver has the task of decoding the code vector $\underline{T}$ from the received vector $\underline{R}$. The algorithm commonly used to perform this decoding operation starts with the computation of a 1-by-(n-k) syndrome vector $\underline{S}$ as:

$$\begin{aligned} \underline{S} &= \underline{R}\ \underline{H}^T \\ &= (\underline{T} \oplus \underline{E})\ \underline{H}^T \\ &= \underline{E}\ \underline{H}^T \end{aligned} \tag{28}$$

The syndrome $\underline{S}$ depends only on the error pattern, and not on the transmitted code word [4]. With syndrome decoding, an (n,k) linear block code can correct up to m errors per code word, provided that n and k satisfy the Hamming bound [4,7];

$$2^{n-k} \geq \sum_{i=0}^{m} \binom{n}{i} \tag{29}$$

where $\binom{n}{i}$ is given by

$$\binom{n}{i} = \frac{n!}{(n-i)!\, i!} \tag{30}$$

There are $2^{n-k}$ syndromes, including the all-zero syndrome. Each syndrome corresponds to a specific error pattern. In general, the syndrome $\underline{S}$ does not uniquely specify the actual error pattern $\underline{E}$. Rather, it identifies the coset that the error pattern belongs to. The most likely error pattern within the coset characterised by the syndrome $\underline{S}$ is the one with the largest probability. Hence the decoding algorithm may be stated as:

1-For the received vector $\underline{R}$, compute $\underline{S} = \underline{R}\ \underline{H}^T$

2-Within the coset characterised by the syndrome $\underline{S}$, choose the error pattern with the largest probability, call it $\hat{\underline{E}}$

3-Compute the code vector

$$\hat{\underline{T}} = \underline{R} \oplus \hat{\underline{E}} \tag{31}$$

as an estimate to the transmitted code word $\underline{T}$.

## VII-APPLICATION EXAMPLE

For example, consider the(7,4) Hamming code with n=7, k=4, and n-k=3. The generator matrix is given as:

$$\underline{G} = \begin{bmatrix} 1\ 0\ 0\ 0\ 1\ 1\ 0 \\ 0\ 1\ 0\ 0\ 0\ 1\ 1 \\ 0\ 0\ 1\ 0\ 1\ 1\ 1 \\ 0\ 0\ 0\ 1\ 1\ 0\ 1 \end{bmatrix} \tag{32}$$

and the parity check matrix is given as:

$$\underline{H} = \begin{bmatrix} 1\ 0\ 1\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ 0\ 1\ 1\ 1\ 0\ 0\ 1 \end{bmatrix} \tag{33}$$

with k = 4, there are $2^4 = 16$ message words which are listed in table I.

Table I, Code words of (7,4) Hamming code

| Message Word $a_1a_2a_3a_4$ | Code Word $a_1a_2a_3a_4c_1c_2c_3$ | Message Word $a_1a_2a_3a_4$ | Code Word $a_1a_2a_3a_4c_1c_2c_3$ |
|---|---|---|---|
| 0000 | 0000000 | 1000 | 1000110 |
| 0001 | 0001101 | 1001 | 1001011 |
| 0010 | 0010111 | 1010 | 1010001 |
| 0011 | 0011010 | 1011 | 1011100 |
| 0100 | 0100011 | 1100 | 1100101 |
| 0101 | 0101110 | 1101 | 1101000 |
| 0110 | 0110100 | 1110 | 1110010 |
| 0111 | 0111001 | 1111 | 1111111 |

Table II Decoding table for (7,4) Hamming Code

| Syndrome $s_1s_2s_3$ | Error Pattern $E_1E_2E_3E_4E_5E_6E_7$ |
|---|---|
| 000 | 0000000 |
| 110 | 1000000 |
| 011 | 0100000 |
| 111 | 0010000 |
| 101 | 0001000 |
| 100 | 0000100 |
| 010 | 0000010 |
| 001 | 0000001 |

The truth tables for the (7,4) Hamming encoder and decoder given in Table I and Table II are now used to derive the power series expansion required for implementation. After some manipulation, the power series expansion for the three bits $c_1$, $c_2$, and $c_3$ in the encoder are given by:

$$c_1 = a_1 + a_3 - 2a_1a_3 - a_2a_3 - a_1a_2 + 3a_1a_2a_3 \tag{34}$$

$$c_2 = a_1 + a_2 + a_4 - 2a_1a_2 - 2a_1a_4 - 2a_2a_4 + 4a_1a_2a_4 \tag{35}$$

$$c_3 = a_1 + a_3 + a_4 - 2a_1a_3 - 2a_1a_4 - 2a_3a_4 + a_1a_2a_3 + 4a_1a_3a_4 - a_1a_2a_3a_4 \tag{36}$$

Similarly, the power series expansions for the syndrome vector and the error pattern vector are expressed as follows:

$$s_1 = r_1r_3 + r_1r_5 - 2r_1r_3r_5 + r_2r_3 + r_2r_5 - 2r_2r_3r_5 - 2r_1r_2r_3 - 2r_1r_2r_5 + 4r_1r_2r_3r_5 \tag{37}$$

$$s_2 = r_1r_4 + r_1r_6 + r_2r_4 + r_2r_6 - 2r_1r_4r_6 - 2r_2r_4r_6 - 2r_1r_2r_4 - 2r_1r_2r_6 + 4r_1r_2r_4r_6 \tag{38}$$

$$s_3 = r_1r_4 + r_1r_7 - 2r_1r_4r_7 + r_3r_4 + r_3r_7 - 2r_3r_4r_7 - 2r_1r_3r_4 - 2r_1r_3r_7 + 4r_1r_3r_4r_7 \tag{39}$$

and

$$E_1 = s_1s_2s_3 \tag{40}$$

$$E_2 = s_1s_2 - s_1s_2s_3 \tag{41}$$

$$E_3 = s_1s_3 - s_1s_2s_3 \tag{42}$$

$$E_4 = s_2s_3 - s_1s_2s_3 \tag{43}$$

$$E_5 = s_1 + s_1s_2s_3 - s_1s_2 - s_1s_3 \tag{44}$$

$$E_6 = s_2 + s_1s_2s_3 - s_1s_2 - s_2s_3 \tag{45}$$

$$E_7 = s_3 + s_1s_2s_3 - s_1s_3 - s_2s_3 \tag{46}$$

Analog implementation for the Hamming (7,4) code encoder at the transmitter side is shown in Fig. 6. Similar implementation for the decoder at the receiver can be designed using Eqs.(37) through (46).

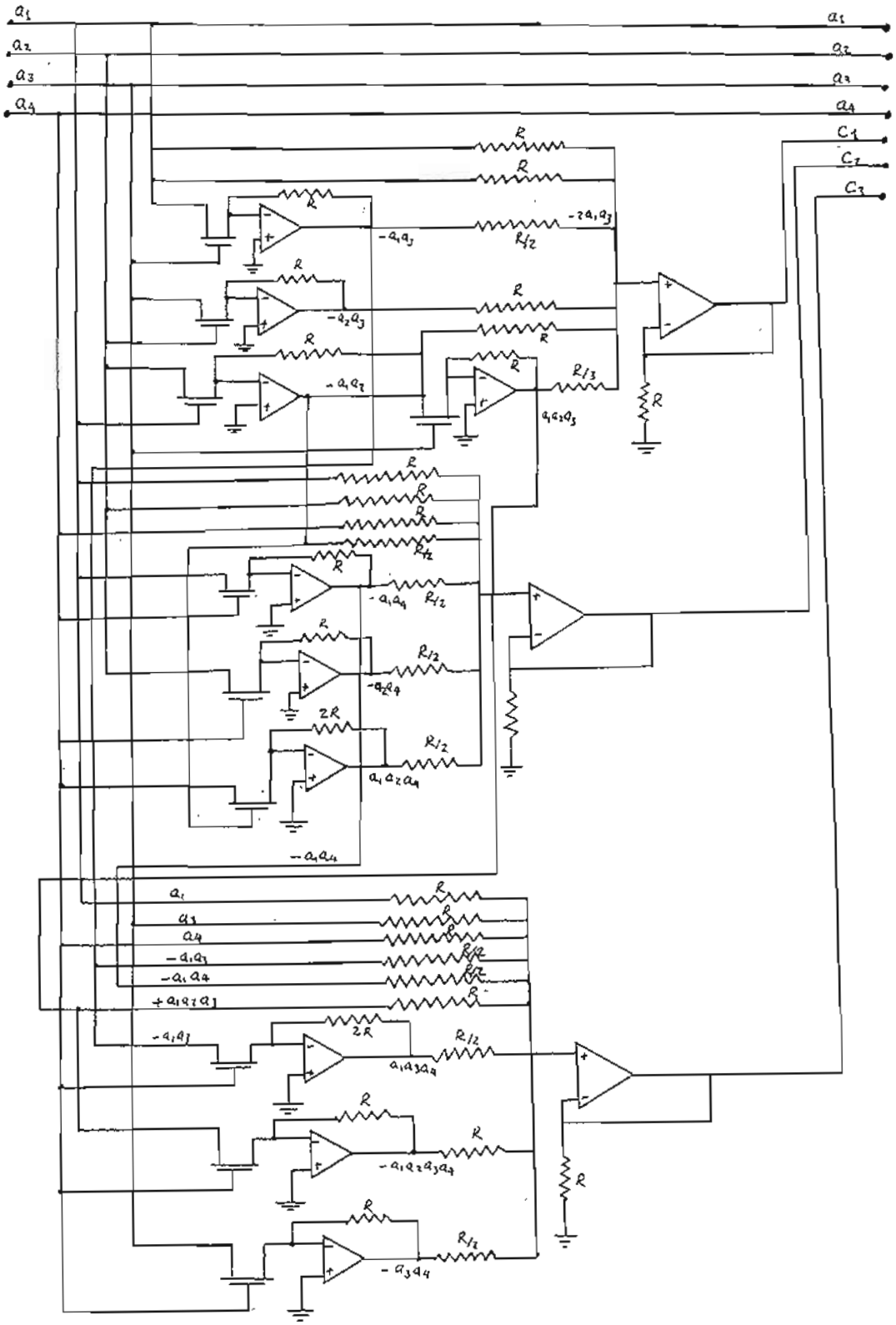


Fig. 6, Analog Implementation of the Hamming (7,4) Encoder.

## VIII-CONCLUSIONS

This paper introduced a new method for representation and implementation of the logical functions. In this method, the truth table is considered as a table of numerical values and curve fitting process was performed to obtain a mathematical relationship between inputs and outputs. A power series expansion method was adopted to simplify the implementation using analog neural networks. The main advantage of the proposed method is the replacement of the classical Karnaugh map reduction technique by a simple mathematical manipulation. Beside, the concept of asynchronous processing is satisfied when the expressions obtained are implemented using neural network structures. With neural networks as the basic building blocks, the problems of fan-in and fan-out associated with the classical logic gates are avoided.

Asynchronous implementation of error detection and error correction encoders/decoders is considered as an application of the proposed method.

## REFERENCES

1-Enab, Yehia M., "An Adaptive Linear Neuron and its Use For Start/Stop Motor Control," Proc. Int. Conf. St., Comp. Science, Soc. and Demographic Research, Ain Shams Univ., Cairo, Egypt, pp409 - 424, March 1990.

2-Tank, D. W., and Hopfield, J. J.,"Simple Neural Optimization Networks: An A/D Converter, a Signal Decision Circuit and a Linear Programming Circuit," IEEE Trans. Circuits and Sys., Vol. CAS-33, No.5, pp533 - 541, May 1986.

3-Widrow, B., and Winter, R., "Neural Nets For Adaptive Filtering and Adaptive Pattern Recognition," IEEE Computer Magazine, pp25 - 39, March 1988.

4-Hamming, R. W., "Coding and Information Theory," Prentice-Hall, 1980.

5-Hamming, R. W., "Error Detecting and Error Correcting Codes," Bell Sys. Tech. J., Vol. 29, ppi47 - 160, April 1950.

6-Popplewell, A., O'Reilly, J. J., and Williams, S., "Architectures For Fast Encoding and Error Detection of Cyclic Codes," IEE Proc. I, Vol. 139, No. 3,pp340 - 348, June 1992.

7-Blahut, R. E., "Theory and Practice of Error Control Codes," Addison-Wesly, New York, 1983.

8-Wu, J., and Shiu, J., "Real Valued Error Control Coding by Using DCT," IEE Proc. I, Vol. 139, No. 2, pp133 - 139, April 1992.

9-Taub, H., and Schilling, D. L., "Principles of Communication Systems," McGraw-Hill, New York, 1987.

10-O'Reilly, J. J., "Series-Parallel Generation of M-Sequences," Radio Electron. Engineer, Vol. 45, pp171 - 176, 1975.

11-O'Reilly, J. J., and Rampaiugul, I., "Circuit Structures for High-Digit-Rate Bit Error Ratio Measurements," IEE Proc. F, Comm., Radar, and Signal Processing, Vol. 134, No. 5, pp434 - 438, Aug. 1987.

12-Khachab, N., and Ismail, M., "A Nonlinear CMOS Analog Cell for VLSI Signal Information Processing," IEEE Journal of Solid-State Circuits, Vol.26, No.11, pp1689 - 1699, Nov. 1991.