ERROR CONTROL BY PRODUCT CODES IN ARITHMETIC UNITS

Ferruccio BARSI and Maria Cristina PINOTTI

Department of Computer Science and Mathematics

University of Perugia, 06123 Perugia, Italy

{barsi,pinotti}@unipg.it

Abstract. *Product (AN) codes constructed in weighted number systems are investigated with the aim of devising error control features suitable for application in arithmetic units. The previous theoretical framework, which was derived in the hypothesis of codes defined in a virtual range $M = k\,A$, is restated for any physical interval $M = b^n$, where b is the radix of the system and n is the number of digits.*

Some general properties holding for radix-b-AN codes are reconsidered and necessary and sufficient conditions for single and double error detection are derived for binary codes and for a sample non binary case. Single error correction is discussed as well and a fast error decoding procedure is suggested and implemented.

Finally, modular AN codes are introduced in order to enable the use of product codes in standard ALU's representing relative integers by means of a radix complement notation. It is shown that the above properties keep their validity and that concurrent single error correction can be associated with arithmetic computation without increasing the time spent for processing.

Index Terms. *Arithmetic codes, arithmetic units, error detection and correction, modular AN codes, product codes, syndrome decoding, weighted AN codes*

1. Introduction.

Arithmetic AN codes have been extensively studied in the past literature in both cases of residue and weighted number system implementations. However, whereas the error detecting and correcting properties of residue AN codes have been completely derived for errors of arbitrary multiplicity [1 - 4], the best results reported in the literature for *weighted AN codes* are limited to particular cases of single error detection and correction [5 - 12]. Moreover, in order to easily preserve the arithmetic closure, weighted AN codes have been studied in a virtual range $M = k\,A$ different from the natural range b^n of a weighted number system. It is easy to verify [9] that the assumption $M = k\,A$ increases the hardware and time costs of code implementation and leads to impractical error detecting and correcting conditions.

In this work, the theory of weighted AN codes is restated for codes ranging in any interval $[0, b^n)$, where b and n indicate the radix and the number of digits of the system, respectively.

Moreover, starting from a new definition of *modular AN codes* it is shown that single and

double error detection or, alternatively, single error correction can be easily obtained in conventional ALU's concurrently with the arithmetic processing.

Unfortunately, the problem of detecting and correcting errors of higher multiplicity remain still unsolved [12 – 17]. However, it is worth noting that the probability of high multiplicity errors is strongly dependent on the mean time between check and, consequently, it is sufficiently small whenever the error control is associated with each computational step.

As this work represents the first attempt of using weighted product codes in practice, no comparison with other solutions is possible.

In what follows, Section 2 reviews the problem of error control in an arithmetic environment together with the fault - error relationships leading to an arithmetic error model. Section 3 derives some simple properties holding for errors of arbitrary multiplicity whereas the problem of single and double error detection is studied in Section 4. In the same Section 4, double error detection in binary systems is analyzed in depth, and error control conditions are presented in a simple and effective form. Non binary codes are also considered for a sample, significant case. Section 5 is devoted to the problem of single error correction and syndrome decoding.

Finally, Sections 6 to 8 are devoted to introduce *modular AN codes* and to extend the results of preceding Sections to enable codes implementation in standard ALU's. Single error correction is presented and it is shown that a correcting procedure can be carried out concurrently with the arithmetic computation.

2. Arithmetic codes

Let’s recall that a code is said to be arithmetic iff it is closed under an arithmetic operation, generally the addition. In other terms, C is an arithmetic code iff, for any two codewords c_1 , c_2 and an arithmetic operator "#", $c_1 \# c_2$ is also a codeword. However, to get more insight into the difference between transmission and arithmetic codes, it is worthwhile to briefly reconsider the fault-error relationships and the corresponding error models.

As usual, assume that an information word is represented as a vector $\underline{X} = (x_{n-1}, x_{n-2}, ..., x_0)$ and suppose that $\underline{X}$ is transmitted through a faulty or noisy channel. Let $\underline{X}^* = (x^*_{n-1}, x^*_{n-2}, ..., x^*_0)$ be the received word. In the hypothesis that a one-to-one correspondence exists between each elementary path of the channel and an information word component, the same correspondence can be assumed between a faulty path in the channel and a wrong information component. Hence, most authors defined the *transmission error* as the *error vector*

$$\underline{E} = \underline{X}^* - \underline{X} = (x^*_{n-1} - x_{n-1}, x^*_{n-2} - x_{n-2}, ..., x^*_0 - x_0)$$

and the *error multiplicity* as the number of non zero error vector components.

Now, assume that $\underline{X}$ and $\underline{Y}$ are two operand to be processed in an arithmetic unit. In this case, the relations between faults and the error components of the result are unpredictable and the error vector becomes a function of the fault and of the actual value of the operands.

In fact, consider, as an example, two pair of binary operands (A = 0 0 1 0 1 1 , B = 0 1 0 0 1 1) and (C = 0 1 1 1 1 1, D = 0 0 0 0 0 1). Adding them pairwise yields S_{AB} = 0 1 1 1 1 0 and S_{CD} = 1 0 0 0 0 0, respectively. If a fault occurs i.e., for instance, the carry from the rightmost position is stacked to "0", adding the above operands will produce the wrong results S^*_{AB} = 0 1 1 1 0 0 and S^*_{CD} = 0 1 1 1 1 0 and, evaluating the corresponding error vectors, it follows that:

$$\underline{E}_{AB} = (0, 0, 0, 0, -1, 0) , \quad \underline{E}_{CD} = (-1, 1, 1, 1, 1, 0)$$

i.e., the same fault will produce two different error vectors.

This conclusion is somewhat surprising. However, recalling that the error ("-1" for the carry in the rightmost bit position) is to be processed in the same arithmetic unit as an additional operand and that the result is defined over the entire information word, it follows that the measure of the error is to be considered at a word level.

In the example, considering the numerical value of the difference between the wrong and the correct results (the *error values*) it follows that, in signed binary notation:

$$E_{AB} = E_{CD} = - 0 0 0 0 1 0$$

and the same conclusion is obtained evaluating the error vectors $\underline{E}_{AB}$, $\underline{E}_{CD}$ as binary, signed digit number representations:

$$E_{AB} = -1 \text{x } 2 = - 2 , \quad E_{CD} = -1 \text{x } 2^5 + 1 \text{x } 2^4 + 1 \text{x } 2^3 + 1 \text{x } 2^2 + 1 \text{x } 2 = - 2$$

i.e.:

$$E_{AB} = E_{CD} = - 0 0 0 0 1 0$$

Previous considerations show that the effect of a fault can be unambiguously evaluated by computing the numerical value of the corresponding error vector.

Evaluating the error multiplicity in arithmetic context is much more difficult than in a transmission one. In fact, for any given error value, there are more equivalent error vectors and the error multiplicity cannot be related to the number of non zero error vector components. The problem is solved by considering *the arithmetic error weight* which is obtained according to the following definitions [10].

Definition 2.1. For any integer N , the arithmetic weight W_{ar} (N) is defined as the smallest number of non zero terms in an expression for N of the form

$$N = \pm a_{n-1} b^{n-1} \pm a_{n-2} b^{n-2} \pm ..... \pm a_0$$

where b is the radix of the system and $0 \leq a_i < b$.

Definition 2.2. The arithmetic distance $D_{ar}(N_1, N_2)$ between two integers N_1 and N_2 is the arithmetic weight of $(N_1 - N_2)$, i.e.

$$D_{ar}(N_1, N_2) = W_{ar}(N_1 - N_2)$$

Definition 2.3. The multiplicity of an arithmetic error E is defined as its arithmetic weight.

The arithmetic distance is a metric. Thus, the error detecting and correcting properties of an arithmetic code C having a minimum code distance D_{ar} are the following:

P1: the code C detects arithmetic errors of multiplicity not greater than t iff $D_{ar} \geq t + 1$

P2: the code C corrects arithmetic errors of multiplicity not greater than t iff $D_{ar} \geq 2t + 1$

P3: the code C detects arithmetic errors up to a multiplicity t_R and, concurrently, corrects arithmetic errors of multiplicity not greater than t_C $(t_R \geq t_C)$ iff $D_{ar} \geq t_R + t_C + 1$

Previous results hold in the hypothesis of codes defined in an unlimited numerical range. In practice, however, numbers and computations are defined in a finite range $M = b^n$, where b is the radix of the number system and n is the number of digits. Applying the above Definition 2.2 in a mod M environment, it follows, in general:

$$D_{ar}(N_1, N_2) = W_{ar}(|N_1 - N_2|_M) \neq W_{ar}(|N_2 - N_1|_M) = D_{ar}(N_2, N_1)$$

and the symmetry of the metric is no longer satisfied.

To overcome this obstacle, most authors restated preceding definitions as follows.

Definition 2.4. For any integer N, the modular weight $W_M(N)$ is given by

$$W_M(N) = \min\{W_{ar}(|N|_M, W_{ar}(|-N|_M)$$

Definition 2.5. The modular distance $D_M(N_1, N_2)$ of two integers N_1 and N_2 is given by $W_M(N_1 - N_2)$.

Definition 2.6. The multiplicity of an arithmetic error in a finite range M is defined as its modular weight.

From Definition 2.4, it is seen that the modular distance satisfies both the positiveness and the symmetry properties of a metric. In addition, since the triangular inequality is satisfied as well [9] provided that, as in our case, $M = b^n$, the modular distance $D_M(N_1, N_2)$ is a metric and, substituting $D_M(N_1, N_2)$ for the arithmetic distance $D_{ar}(N_1, N_2)$, the arithmetic error detecting and correcting properties *P1 – P3* keep their validity.

3. AN codes in weighted number systems

A product or AN code is defined as a code representing any integer N by the product AN where A is an integer called the generator of the code. From the definition, it is immediately verified that:

Theorem 3.1. Given an AN code C, any error E affecting a code word is detected iff $E \not\equiv 0 \bmod A$.

In what follows, attention will be focused on product codes implemented in weighted number systems of fixed radix b. Moreover, in order to minimise the code redundancy, it will be assumed that $(A, b) = 1$. The next two conditions hold whose proofs are given in Appendix.

Theorem 3.2. A necessary condition for an AN code C be t - detecting is

Theorem 3.3. An AN code of generator is t - detecting for $t = 1$ and $t = n - 1$, where n indicates the number of digits of the representation.

4. Single and Double Error Detection

Consider a weighted system with n digits and radix b and let $E = e_i b^i$ be a single error. Then a product code C will detect E iff

$$e_i b^i \not\equiv 0 \bmod A$$

for arbitrary $e_i \neq 0$, $i = 0, \ldots, n-1$, $-b < e_i < b$.

Recalling that it has been assumed that $(A, b) = 1$, it is trivially derived that C is single error detecting iff $e_i \not\equiv 0 \bmod A$. Then, the following necessary and sufficient condition can be easily proved.

Theorem 4.1. An AN code constructed in a weighted system of radix b with $(A, b) = 1$ is 1-detecting iff $A \geq b + 1$.

Proof. Necessity. To prove necessity, assume that C is an 1-detecting AN code. Then $e_i \not\equiv 0 \bmod A$ for any non zero e_i, $-b < e_i < b$, i.e., necessarily, $A > \max |e_i| = b - 1$. Recalling that $(A, b) = 1$, it follows that $A \geq b + 1$ and the first part of the theorem is proved.

Sufficiency. To prove sufficiency, assume that $A \geq b + 1$. However, this assumption implies $e_i \not\equiv 0 \bmod A$ for any non zero e_i, $-b < e_i < b$ and the proof is completed.

Dealing with double error detection is more difficult than detecting single errors.

A double error E is an error which can be expressed in the form:

$$E = e_i b^i + e_j b^j$$

and an AN code will be 2 - detecting iff $e_i b^i + e_j b^j \not\equiv 0 \bmod A$ for any pair $e_i , e_j \neq 0$, $i, j = 0, ... (n - 1), i \neq j , - b < e_i , e_j < b$.

Recalling the assumption $(A, b) = 1$ and letting, without loss of generality, $i < j$, the above condition can be restated in the form $e_i + e_j b^{j-i} \not\equiv 0 \bmod A$ or, with a slightly different notation:

$$e_i + e_j b^k \not\equiv 0 \bmod A$$

where $k = 1, ... , n - 1, e_i , e_j \neq 0, - b < e_i , e_j < b , i, j = 0, ... (n - 1) , i \neq j$.

As the following error detecting conditions will be derived by using number theory tools and concepts, some useful definitions and properties will be reported for the sake of completeness.

Definition 4.1. For any integer $m \neq 0$, the Euler function $\phi (m)$ is the number of positive integers $x, x < |m|$, such that $(x, m) = 1$.

Observing that $\phi (m) = \phi (- m)$, it follows that attention can be limited to positive integers.

Theorem 4.2. (Euler-Fermat theorem) Given two integers a and $m > 1$ such that $(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \bmod m$.

Definition 4.2. Given an integer a prime to a positive integer $m > 1$, the least positive integer d_m such that

$$a^{d_m} \equiv 1 \bmod m$$

is called the exponent to which a belongs mod m.

As an example, consider integers - 1, 2 and 3 and let $m = 7$. Then the exponents to which the above integers belong mod 7 are 2, 3 and 6, respectively.

Theorem 4.3. Given an integer a prime to a positive integer $m > 1$, let d_m be the exponent to which a belongs mod m. Then, for positive integers s and t :

$$a^s \equiv a^t \bmod m$$

iff:

$$s \equiv t \bmod d_m$$

Proof. Necessity. Assume, without loss of generality, $s \geq t$. The difference $(s - t)$ can be

expressed as:

$$s - t = d_m q + r, \quad 0 \leq r < d_m$$

As $(a, m) = 1$, from the congruence $a^s \equiv a^t \bmod m$ it follows that

$$1 \equiv a^{(s-t)} \equiv (a^{d_m})^q \, a^r \bmod m$$

and, necessarily, $r = 0$, i.e., $s \equiv t \bmod d_m$.

Sufficiency. Let $s \equiv t \bmod d_m$. Then $s = t + k\, d_m$ and, trivially, $a^s \equiv a^t \bmod m$.

Corollary 4.1. Given an integer a prime to a positive integer $m > 1$, the integers

$$a, a^2, a^3, \ldots, a^{d_m}$$

are distinct mod m.

Proof. For any pair of positive integers (i, j), $i \neq j$, $1 \leq i, j \leq d_m$, the congruence $a^i \equiv a^j \bmod m$ cannot be verified. To prove this, suppose the contrary. Then, from Theorem 4.3, $a^i \equiv a^j \bmod m$ would imply $i \equiv j \bmod d_m$, i.e., d_m would divide $(i - j)$ and a contradiction arises.

Theorem 4.4. Given an integer a prime to a positive integer $m > 1$, if, for a positive integer τ, $1 \leq \tau < d_m$

$$a^\tau \equiv -1 \bmod m$$

then, necessarily

$$\tau = d_m / 2$$

Proof. From the congruence

$$a^\tau \equiv -1 \bmod m$$

it follows that

$$a^{2\tau} \equiv 1 \bmod m$$

or, from Theorem 4.3, $2\tau = k\, d_m$ and, recalling that $\tau < d_m$, $k = 1$ and hence $\tau = d_m / 2$.

Corollary 4.2. In the hypothesis of Theorem 4.4, d_m is even.

Theorem 4.5. Given an integer a prime to a positive integer $m > 1$, with d_m even, if

$$a^{d_m/2} \equiv -1 \bmod m$$

then, necessarily, for any non negative integer h :

$$a^h \equiv -a^{h + d_m/2} \bmod m$$

Proof. The theorem is trivially proved multiplying for a^h both sides of the congruence

$$a^{d_m/2} \equiv -1 \bmod m$$

Theorem 4.6. Given an integer a prime to a positive integer $m > 1$, if the congruence $a^{\tau} \equiv -1 \bmod m$ is never verified for a positive integer τ, $1 \leq \tau < d_m$, then a^h, where h is any non negative integer, has no additive inverse of the form a^k, $k \geq 0$ in the ring mod m.

Proof. By contradiction, suppose that the congruence

$$a^h \equiv -a^k \bmod m$$

is verified for two non negative integers h and k. Assuming, without loss of generality, $h > k$ and dividing both sides of the congruence by a^k, it is obtained

$$a^{h-k} \equiv -1 \bmod m$$

which contradicts the original hypothesis.

The following Corollary 4.3 is immediate.

Corollary 4.3. Given an integer a prime to a positive integer $m > 1$, if, for two non negative integers h and k :

$$a^h \equiv -a^k \bmod m$$

then

$$a^{d_m/2} \equiv -1 \bmod m$$

Theorem 4.7. Given an integer a prime to a positive integer $m > 1$, the multiplicative inverse of any a^h in the mod m ring is a^k with

$$k \equiv -h \bmod d_m$$

where h and k are non negative integers.

Proof. Observe first that the existence of the multiplicative inverse of $a^h \bmod m$ is guaranteed as $(a, m) = 1$ implies $(a^h, m) = 1$. Moreover, by definition of the multiplicative inverse :

Then, dividing both sides of the congruence by a^{h^*}, where $0 \leq h^* \leq d_m$, $h^* \equiv h \bmod d_m$:

It is concluded that, in general, the multiplicative inverse of a^h will be expressed as a^k with $k \equiv d_m - h^* \equiv -h \bmod d_m$.

Definition 4.2. Let m be a positive integer. Then, any integer h such that the exponent to which it belongs mod m is $\phi(m)$ is called a primitive root of m.

Now, we are ready to prove two necessary and sufficient conditions for a weighted AN code to be 2-detecting (see Appendix for proofs).

The most significant case concerns binary systems for which the general condition for double error detection

$$e_i + e_j b^k \not\equiv 0 \bmod A$$

can be restated in the simpler form:

$$2^k \not\equiv \pm 1 \bmod A$$

with $k = 1, \ldots, n-1$.

Theorem 4.8. A binary AN code with $(A, 2) = 1$ is 2 - detecting iff

- $d_A \geq n$ if d_A is odd or d_A is even with $2^{d_A/2} \not\equiv -1 \bmod A$
- $d_A \geq 2n$ if d_A is even with $2^{d_A/2} \equiv -1 \bmod A$

where d_A represents the exponent to which 2 belongs mod A and n is the number of bits of the code.

As an example, the following Table I shows the least generators A for some increasing values of n, as derived from Theorem 4.8.

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A_{min}	7	11	11	13	19	19	19	23	23	29	29	29	37	37	37	37	47	47
d_A	3	10	10	12	18	18	18	11	11	28	28	28	36	36	36	36	23	23

The validity of Theorem 4.8 is limited to the case of binary systems. A general formulation holding for arbitrary radix b is still unknown. However, it is possible to derive necessary and sufficient conditions for double error detection if some additional assumptions are made. To this purpose, assume that the code generator A is a prime and that the radix b is a primitive root of A. In these hypotheses, the following Theorem holds.

Theorem 4.9. A weighted AN code of radix b where the generator A is a prime and b is a primitive root of A is 2 - detecting iff

$$\delta \geq n$$

where n indicates the number of digits and

where and k_j ($j = u, v$) is an integer in [1, A - 1] such that $b^{k_j} \equiv j \mod A$.

5. Single Error Correction

Some general conditions leading to double error detection have been presented in preceding Section 4. From the properties of the arithmetic metric, it is concluded that a double error detecting code can be used, alternatively, for single error correction. To do this, let's define first an error figure according to the following definition.

Definition 5.1. Given a product code of generator A and any integer X to be checked, the number $S = |X|_A$ will be defined as the syndrome of X.

Now, suppose that a code C satisfies the conditions for single error correction and assume that a single error $E = e_i b^i$, $i = 0, 1, ..., n-1$, $-b < e_i < b$ affects a legitimate code word X thus generating a wrong number $X' = X + E$. To perform correction, the syndrome $S = |X'|_A$ is first computed and:

$$S \equiv X' = X + E \equiv E = e_i b^i \mod A$$

As the code C is single error correcting, multiplying (in parallel) the above congruence by the (n - 1) multiplicative inverses $1/|b^i| \mod A$, there will be one and only one value

$$s_i = S \times 1/|b^i| \equiv e_i \mod A$$

where $i = 0, 1, ... , n - 1$ and $-b < e_i < b$. In other terms, recalling preceding limitations, there will be a unique integer s_i such that

- s_i ranges in [1, b) i.e., $e_i = s_i$ or, alternatively:
- s_i ranges in [$A - b - 1$, A) i.e., $e_i = s_i - A$.

and the correct number will be reconstructed as

$$X = X' - E = X' - e_i b^i$$

As an example of the application of the procedure, consider a binary product code of generator $A = 19$ with $n = 7$, $d_A = 18$, satisfying conditions of Theorem 4.8. Computing the multiplicative inverses of $2^i \mod A$, $i = 0, ... , 6$ it is obtained:

$$1/2^0 \mod A = 1$$
$$1/2^1 \mod A = 10$$
$$1/2^2 \mod A = 5$$
$$1/2^3 \mod A = 12$$
$$1/2^4 \mod A = 6$$
$$1/2^5 \mod A = 3$$

$$1/2^{6} \bmod A = 11$$

Now, let

$$X = 0\,1\,0\,1\,0\,0\,0_2 = 38_{10}$$

be a legitimate codeword and suppose that a single error

$$E = -\ 0\,0\,1\,0\,0\,0\,0_2 = -\ 1 \times 2^{4} = -\ 16_{10}$$

affects integer X thus yielding:

$$X' = X + E = 0\,0\,1\,0\,1\,1\,0_2 = 22_{10}$$

Computing the syndrome of X', it is obtained:

$$S = |X'|_A = |22|_{19} = 3_{10}$$

and, multiplying the syndrome by the multiplicative inverses of 2^i, I = 1, 2, ,… , 6 it is derived:

$$s_0 = 3_{10}$$
$$s_1 = |S \times 1/2|_A = 11_{10}$$
$$s_2 = |S \times 1/2^{2}|_A = 15_{10}$$
$$s_3 = |S \times 1/2^{3}|_A = 17_{10}$$
$$s_4 = |S \times 1/2^{4}|_A = 18_{10}$$
$$s_5 = |S \times 1/2^{5}|_A = 9_{10}$$
$$s_6 = |S \times 1/2^{6}|_A = 14_{10}$$

Here, only one s_i, namely $s_4 = 18_{10}$ belongs to the legitimate error range [1, b), [$A - b$ - 1, A) and, consequently, $e_i = s_i - A. = 18 - 19 = -1$. The error is then reconstructed as $E = -2^4$ and the correct value of X becomes:

$$X = X' - E = 22_{10} - (-16_{10}) = 38_{10}$$

6. Using AN codes in arithmetic units

In Section 3, a product code has been defined as a code representing any integer N by the product AN for some suitable constant integer A, the generator of the code. The arithmetic properties of such a code derive from its definition. In fact, assuming that two integers X and Y are encoded in a product code and then added together, it is obtained:

$$A\,X + A\,Y = A\,(X + Y)$$

i.e., adding two code words produces another code word.

In practice, however, integers are defined in a finite range $M = b^n \neq k\,A$ (where b and n indicate the radix and the number of digits of the system, respectively), relative integers are represented in a radix complement notation and computations are carried out mod M. Consequently, the representation $|A\,Z|_M$ of code words corresponding to relative integers Z or to the sum of two code words $|A\,X + A\,Y|_M$ may produce results which are no more code words.

In the past literature, most authors solved this problem by limiting their consideration to non negative integers and by assuming a code range $M = k\,A$, to be implemented in a physical range $b^n \geq M$.

However, a direct approach to *AN* codes constructed in standard arithmetic units is possible. In fact, suppose that integers X in the range $[-m, m)$ are considered and represented by means of a product code of generator A. The code words $X_A = A\,X$ will range in $[-A\,m, A\,m)$ and, representing by means of a complement notation in the range $M = b^n$, $2\,A\,m \leq M < 2\,A\,(m+1)$ it will be obtained:

$$\boldsymbol{X} = |X_A|_M = |A\,X|_M$$

or, recalling the complement notation protocol:

$$\boldsymbol{X} \equiv 0 \bmod A \text{ if } 0 \leq X < m \text{, i.e., } 0 \leq |X_A|_M = X_A < M/2$$
$$M - \boldsymbol{X} \equiv 0 \bmod A \text{ if } -m \leq X < 0 \text{, i.e., } M/2 \leq |X_A|_M = M + X_A < M$$

Now, suppose that the representations of two legitimate code words $\boldsymbol{X} = |X_A|_M = |A\,X|_M$ and $\boldsymbol{Y} = |Y_A|_M = |A\,Y|_M$ are added and let $\boldsymbol{S} = |\boldsymbol{X} + \boldsymbol{Y}|_M$ be the representation of their sum. Once again, recalling the basic elements of computer arithmetic, it is concluded that

$$\boldsymbol{S} \equiv 0 \bmod A \text{ if } 0 \leq \boldsymbol{S} < M/2$$
$$M - \boldsymbol{S} \equiv 0 \bmod A \text{ if } M/2 \leq \boldsymbol{S} < M$$

in the absence of arithmetic overflow and

$$\boldsymbol{S} \equiv 0 \bmod A \text{ if } M/2 \leq \boldsymbol{S} < M$$
$$M - \boldsymbol{S} \equiv 0 \bmod A \text{ if } 0 \leq \boldsymbol{S} < M/2$$

if overflow occurred.

Conversely, any representation $\boldsymbol{Z}$ can be recognized to be legitimate (i.e., corresponding to the representation of a code word or to the correct sum of two code words) if:

$$\boldsymbol{Z} \equiv 0 \bmod A \text{ and } 0 \leq \boldsymbol{Z} < \mathrm{M}/2$$
$$M - \boldsymbol{Z} \equiv 0 \bmod A \text{ and } M/2 \leq \boldsymbol{Z} < M$$

whereas, if an overflow is detected:

$$\boldsymbol{Z} \equiv 0 \bmod A \text{ and } M/2 \leq \boldsymbol{Z} < M$$
$$M - \boldsymbol{Z} \equiv 0 \bmod A \text{ and } 0 \leq \boldsymbol{Z} < M/2$$

Preceding considerations define *modular AN codes* in conventional ALU's and prove that the arithmetic properties of product codes are preserved according to the following definitions and properties.

Definition 6.1. In a **modular A N code** representing integers in $[-m, m)$ in the range $M = b^n$, $2\,A\,m \leq M < 2\,A\,(m+1)$, by means of a complement notation, any integer $\boldsymbol{X}$, $0 \leq$

$\boldsymbol{X} < M$, will be recognized to be a code word iff:

$$\boldsymbol{X} \equiv 0 \bmod A \text{ if } 0 \leq \boldsymbol{X} < M/2$$
$$M - \boldsymbol{X} \equiv 0 \bmod A \text{ if } M/2 \leq \boldsymbol{X} < M$$

Property 6.1. In the absence of an arithmetic overflow, the sum $\boldsymbol{S} = |\boldsymbol{X} + \boldsymbol{Y}|_M$ of any two code words $\boldsymbol{X}$ and $\boldsymbol{Y}$ is also a code word.

Property 6.2. Let $\boldsymbol{S} = |\boldsymbol{X} + \boldsymbol{Y}|_M$ be the sum of any two code words $\boldsymbol{X}$ and $\boldsymbol{Y}$. Then, if an arithmetic overflow occurs

$$\boldsymbol{S} \equiv 0 \bmod A \text{ if } M/2 \leq \boldsymbol{S} < M$$
$$M - \boldsymbol{S} \equiv 0 \bmod A \text{ if } 0 \leq \boldsymbol{S} < M/2$$

Definition 6.2. In a **modular A N code** representing integers in $[-m, m)$ in the range $M = b^n$, $2Am \leq M < 2A(m+1)$, by means of a complement notation, any integer $\boldsymbol{Z}$, $0 \leq \boldsymbol{Z} < M$, will be referred to as a legitimate representation if

$$\boldsymbol{Z} \equiv 0 \bmod A \text{ or } \boldsymbol{Z} \equiv M \bmod A$$

Now, let $\boldsymbol{X}$ be a codeword or the sum of any two codewords and suppose that an error E, $-M < E < M$, affects the legitimate information $\boldsymbol{X}$ thus generating in $[0, M)$ the wrong number $\boldsymbol{X}^*$:

$$\boldsymbol{X}^* = \boldsymbol{X} + E$$

In order to devise the conditions for detecting E, consider the following two cases for $\boldsymbol{X}$.

Case 1

$\boldsymbol{X} \equiv 0 \bmod A$. In this case, the error will be detected iff

$$\boldsymbol{X}^* \not\equiv 0 \bmod A \quad \text{i.e., } E \not\equiv 0 \bmod A$$

and

$$\boldsymbol{X}^* \not\equiv M \bmod A \quad \text{i.e., } E \not\equiv M \bmod A$$

Case 2

$\boldsymbol{X} \equiv M \bmod A$. In this case, the error will be detected iff

$$\boldsymbol{X}^* \not\equiv 0 \bmod A \quad \text{i.e., } E \not\equiv -M \bmod A$$

and

$$\boldsymbol{X}^* \not\equiv M \bmod A \quad \text{i.e., } E \not\equiv 0 \bmod A$$

However, as far as errors E are considered in the range $(-M, M)$, to each E in the range $(0, M)$, there corresponds a mod M equivalent $E^* = -(M - E)$ in $(-M, 0)$ for which

$$E^* \equiv E \bmod M$$

and, conversely, to each error in the range (- M , 0), there corresponds in (0, M) a mod M equivalent error $E^* = E + M$ for which

$$E^* \equiv E \mod M$$

Now, observing that two errors which are mod M equivalent have the same modular weight, consider a pair $\{E , E^* \}$ and suppose, without loss of generality, that $E > 0$.

Then, if $E \not\equiv 0 \mod A$, il follows $E^* \not\equiv -M \mod A$ and, if $E \not\equiv M \mod A$, il follows $E^* \not\equiv 0 \mod A$.

Preceding considerations prove the following Theorem 3.1.

Theorem 6.1. Given a modular AN code C, any error corresponding to a mod M equivalent pair $\{E , E^* \}$, $-M < E, E^* < M$ affecting a code word or the sum of two code words will be detected iff

$$E \not\equiv 0 \mod A$$

and

$$E^* \not\equiv 0 \mod A$$

7. Error detection and correction in arithmetic units

In a mod M arithmetic unit, $M = b^n$ (where b and n indicate the radix and the number of digits of the system), let E be a single error in the range ($-M$, M) for which, according to Definition 2.6:

$$|E|_M = \varepsilon_i \, b^i \quad \text{or} \quad |-E|_M = \varepsilon_i \, b^i$$

i.e.,

$$E = \pm \varepsilon_i \, b^i \quad , \; 0 < \varepsilon_i < b$$

or, equivalently:

$$E = e_i \, b^i$$

with $-b < e_i < b$, $i = 0, ... , n - 1$.

From Theorem 6.1, observing that $|E^*|_M = |E|_M$, it is concluded that a single error is detected iff

$$e_i \, b^i \not\equiv 0 \mod A \quad \text{for any } e_i \neq 0, \; i = 0, ... , n -1 , \; -b < e_i < b .$$

As a consequence, Theorem 4.1 can be restated as follows.

Theorem 7.1. A modular AN code constructed in a weighted system of radix b with (A, b) = 1 is 1-detecting iff $A \geq b + 1$.

Similarly, consider a double error E in the range (- M, M) for which, according to Definition 2.6:

$$|E|_M = \varepsilon_i b^i \pm \varepsilon_j b^j \quad \text{or} \quad |-E|_M = \varepsilon_i b^i \pm \varepsilon_j b^j$$

with $0 < \varepsilon_i , \varepsilon_j < b$, $0 < \varepsilon_i b^i \pm \varepsilon_j b^j < M$ or, equivalently:

$$E = e_i b^i + e_j b^j$$

for $- b < e_i , e_j < b$, $i, j = 0, ... (n -1)$, $i \neq j$, $e_i , e_j \neq 0$.

Once again, observing that $|E *|_M = |E|_M$, it is concluded from Theorem 6.1 that a modular AN code is 2 - detecting iff $e_i b^i + e_j b^j \not\equiv 0 \bmod A$. Recalling the assumption (A , b) = 1 and letting, without any loss of generality, $i < j$, the 2 - detecting condition can be restated in the form:

$$e_i + e_j b^k \not\equiv 0$$

where $k = 1, ... , n -1$, $e_i , e_j \neq 0$, - b $< e_i , e_j < b$, $i, j = 0, ... (n -1)$, $i \neq j$.

It is immediate to realise that the results of Theorems 4.8 and 4.9 apply to modular AN codes. In particular, Theorem 4.8 takes the form:

Theorem 7.2. A binary modular AN code with (A , 2) = 1 detects double errors or, alternatively, corrects single errors affecting a codeword or the sum of any two codewords iff

- $d_A \geq n$ if d_A is odd or d_A is even with $2^{d_A/2} \not\equiv - 1 \bmod A$
- $d_A \geq 2n$ if d_A is even with $2^{d_A/2} \equiv - 1 \bmod A$

where d_A represents the exponent to which 2 belongs mod A and n is the number of the bits of the code.

8. Single Error Correction in arithmetic units

In this Section, the results reported in Section 5 will be extended to consider single error correction in standard ALU's.

To this purpose, the definition of the syndrome will slightly modified to consider the number representations, i.e., the non negative integers $\boldsymbol{X} = |A X|_M$:

$$S = |\boldsymbol{X}|_A$$

And, from Definition 6.1 and Properties 6.1 and 6.2, $\boldsymbol{X}$ will be recognized to be legitimate iff:

$$S = 0 \text{ or } S = |M|_A$$

Now, suppose that a code C satisfies conditions of Theorem 7.2 and assume that a single error $E = e_i b^i$, $i = 0, 1, ..., n-1$, $-b < e_i < b$, affects a legitimate representation $\boldsymbol{X}$ thus generating a wrong number $\boldsymbol{X}' = \boldsymbol{X} + E$. Then, for the syndrome S :

$$S \equiv e_i b^i \bmod A \quad \text{or} \quad S \equiv M + e_i b^i \bmod A$$

As the code C is single error correcting, multiplying the above congruences, in parallel, by the $(n - 1)$ multiplicative inverses $1/|b^i| \bmod A$, there will be one and only one value s_i for which

$$s_i = |S \times 1/|b^i||_A \equiv e_i \bmod A$$

or

$$s_i = |S \times 1/|b^i||_A \equiv M + e_i \bmod A$$

with $i = 0, 1, ... , n - 1$ and $-b < e_i < b$. In other terms, there will be a unique s_i such that:

- s_i ranges in $[1, b)$ and $e_i = s_i$ or in $[A - b + 1, A)$ and $e_i = s_i - A$

or, alternatively:

- s_i is in $(|M|_A, |M + b|_A)$ and $e_i = |s_i - M|_A$ or in $(|M - b|_A, |M|_A)$ and $e_i = -|s^*_i - M|_A$.

and the correct number will be reconstructed as

$$X = X' - E = X' - e_i b^i$$

As an example, consider a binary product code of generator $A = 23$ with $n = 11$, $d_A = 11$, $M = 2048$ representing integers in the range $[-1024, 1024)$ and satisfying the conditions of Theorem 7.2. In this system, the multiplicative inverses of $2^i \bmod A$, $i = 0, ... , 10$ are:

$$1/2^0 \bmod A = 1$$
$$1/2^1 \bmod A = 12$$
$$1/2^2 \bmod A = 6$$
$$1/2^3 \bmod A = 3$$
$$1/2^4 \bmod A = 13$$
$$1/2^5 \bmod A = 18$$
$$1/2^6 \bmod A = 9$$
$$1/2^7 \bmod A = 16$$
$$1/2^8 \bmod A = 8$$
$$1/2^9 \bmod A = 4$$
$$1/2^{10} \bmod A = 2$$

Now, consider two code words $\boldsymbol{X} = 1\,0\,1\,0\,1\,1\,1\,1\,1\,0\,0 = 1404_{10}$ and $\boldsymbol{Y} = 0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,1 = 989_{10}$ for which $\boldsymbol{X} \equiv 0 \bmod A$ and $\boldsymbol{Y} \equiv M \bmod A$ and let $\boldsymbol{Z}$ be their sum

$$\boldsymbol{Z} = |\boldsymbol{X} + \boldsymbol{Y}|_M = 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1 = 345_{10}.$$

If a single error $E = -2^5 = 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0 = -32_{10}$ affects $\boldsymbol{Z}$, the wrong result $\boldsymbol{Z}'' = 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1 = 313_{10}$ will be generated with syndrome $\boldsymbol{S} = |313|_{23} = 14$.

Multiplying $\boldsymbol{S}$ by the multiplicative inverses of $2^i \bmod A$, $i = 0, \ldots, 10$, it is derived:

$$s_0 = |14 \times 1|_{23} = 14$$
$$s_1 = |14 \times 12|_{23} = 7$$
$$s_2 = |14 \times 6|_{23} = 15$$
$$s_3 = |14 \times 3|_{23} = 19$$
$$s_4 = |14 \times 13|_{23} = 21$$
$$s_5 = |14 \times 18|_{23} = 22$$
$$s_6 = |14 \times 9|_{23} = 11$$
$$s_7 = |14 \times 16|_{23} = 17$$
$$s_8 = |14 \times 8|_{23} = 20$$
$$s_9 = |14 \times 4|_{23} = 10$$
$$s_6 = |14 \times 2|_{23} = 5$$

Only one s_i, namely $s_5 = 22_{10}$ satisfies the conditions of the procedure ($e_5 = -1$) and the correct value of $\boldsymbol{Z}$ becomes:

$$\boldsymbol{Z} = \boldsymbol{X}' - E = |313_{10} + 25_{10}|_M = 345_{10}$$

It is worth noting that, besides correcting errors, the procedure enables concurrent detection of additive overflow. In fact, it can be applied to recover a legitimate representation which can be finally checked to see if the representation is also a codeword.

Appendix

Proof of Theorem 3.2

Proof. For an AN code be *t*-detecting, it is necessary that the arithmetic weight of any multiple of the code generator A satisfies the inequality $W_{ar}(k\,A) \geq t + 1$. To prove this, suppose, by contradiction, that $W_{ar}(k^*\,A) < t + 1$. Then, there would exist a multiple of A :

where $\tau \leq t$, and, consequently, the error $E = k^*\,A$ of multiplicity not greater than t could not be detected.

Observing that the Hamming (and, necessarily, the arithmetic) weight of any generator is at most t , the theorem follows.

Proof of Theorem 3.3

Proof. Before proving the theorem, observe that $(b^i, \sum_{j=0}^{s} b^j) = 1$ for arbitrary i and s..

Showing that the theorem holds for $t = 1$ is immediate as, for any single error $E = e_i\,b^i$, the congruence:

$$e_i\,b^i \equiv 0 \bmod (b+1)$$

is never verified because $(b^i, 1 + b) = 1$ and $-b < e_i < b$.

To prove the last part of the theorem, assume $t = n - 1$ and let E be an error of multiplicity t :

where $-b < e_{ij} < b$, $e_{ji} \neq 0$, $0 \leq ij \leq n - 1$. As $n = t + 1$, it follows that:

$$-b^n + b \leq E \leq b^n - b$$

Assuming, by contradiction, that the code is not t - detecting implies that there exists at least an undetectable error

with $k \neq 0$. Then, recalling the range of the error E , it is obtained:

$$|k| \leq \left\lfloor \frac{b^n - b}{b^{t+1} - 1}(b-1) \right\rfloor = \left\lfloor \frac{\max|E|}{A} \right\rfloor < b - 1$$

Now, observe that has $(t+1) = n$ terms, each consisting of a power of b , whereas E has only $(n - 1)$ similar terms. Let b^u be the lacking power of b in E, with $u \neq 0$.

Recalling preceding equality:

E

and, necessarily:

As ranges in and $\sum_{i=0}^{u} b^i = \frac{b^{u+1}-1}{b-1} < b-1$, the assumption $h = 0$ would imply $k = 0$, i.e., no error.

On the other hand, the complementary assumption $|h| \geq 1$ would imply that preceding equality could not be verified. In fact, recalling that $|k| \leq b - 2$:

or, as :

To complete the proof, consider the case where $u = 0$. It follows that:

and a contradiction arises as $k \leq b - 2$ cannot be a multiple of b.

Proof of Theorem 4.8

Proof. Necessity. If the code is 2 - detecting, the following congruence will never be verified:

$$2^{\tau} \equiv \pm 1 \bmod A \text{ , } 1 \leq \tau \leq n - 1$$

As the congruence:

$$2^{\tau} \equiv + 1 \bmod A$$

has a solution for $\tau = k\ d_A$, k being a non negative integer, then, necessarily:

$$\max \tau < d_A$$

or, equivalently:

$$n \leq d_A$$

Similarly, the complementary congruence:

$$2^{\tau} \equiv - 1 \bmod A$$

is not to be verified in the hypothesis of a 2-detecting code. On the other hand, in general, this congruence can be satisfied only if d_A is even and, if a solution exists, it is of the form

$$\tau = (2k + 1)\ d_A/2$$

Hence, once again, necessarily:

$$\max \tau < d_A/2$$

or, equivalently:

$$n \leq d_A/2$$

Sufficiency. Suppose that the condition of the theorem holds and assume, by contradiction, that the code is not 2 - detecting. Then, there will exist an integer τ in the range [1, n - 1] such that:

$$2^{\tau} \equiv \pm 1 \bmod A$$

However, the congruences $2^{\tau} \equiv 1 \bmod A$ and $2^{\tau} \equiv -1 \bmod A$ imply, respectively, $\tau = k\, d_A$ and $\tau = (2k + 1)\, d_A/2$ and a contradiction follows.

Proof of Theorem 4.9

Proof. Before proving the theorem, observe first that, as b is a primitive root of A and A is a prime, then $d_A = \phi(A) = A - 1$. This implies that, for any j in $[1, A - 1]$ there exists an integer k_j in the same range $[1, A - 1]$ for which $b^{k_j} \equiv j \bmod A$.

Necessity. If the code is 2 - detecting, the congruence

$$e_i\, b^{\tau} \equiv \pm e_j \bmod A$$

will never hold for every τ ranging in $[1, n - 1]$ and $i, j = 0, ...,. n - 1$, $i \neq j$, $-b < e_i, e_j < b$. Referring to the absolute error values $\varepsilon_i = |e_i|$ and $\varepsilon_j = |e_j|$, preceding congruence is equivalent to the following:

$$\varepsilon_i\, b^{\tau} \equiv \pm \varepsilon_j \bmod A$$

with $\varepsilon_i, \varepsilon_j$ in $[1, b - 1]$ and τ ranging again in $[1, n - 1]$.

Now, let $k_{\varepsilon i}$ and $k_{\varepsilon j}$ be two integers in $[1, A - 1]$ such that $b^{k_{\varepsilon i}} \equiv \varepsilon_i \bmod A$ and $b^{k_{\varepsilon j}} \equiv \varepsilon_j \bmod A$. Substituting for $\varepsilon_i, \varepsilon_j$ in the above congruences yields:

$$b^{k_{\varepsilon i}}\, b^{\tau} \equiv b^{k_{\varepsilon j}} \bmod A \qquad (4.9.1)$$

$$b^{k_{\varepsilon i}}\, b^{\tau} \equiv -b^{k_{\varepsilon j}} \bmod A \qquad (4.9.2)$$

From Theorem 4.3 it follows that Congruence (4.9.1) is verified iff

$$\tau \equiv (k_{\varepsilon j} - k_{\varepsilon i}) \bmod d_A \qquad (4.9.1')$$

Similarly, recalling Theorem 4.5, Congruence (4.9.2) will be restated as

$$b^{k_{\varepsilon i}}\, b^{\tau} \equiv b^{k_{\varepsilon j} + d_A/2} \bmod A$$

and, from the same Theorem 4.3 , Congruence (4.9.2) will be verified iff

$$\tau \equiv (k_{\varepsilon j} - k_{\varepsilon i}) + d_A/2 \bmod d_A \qquad (4.9.2')$$

As a conclusion, the code will be 2 - detecting provided that

$$\tau \equiv (k_{\varepsilon j} - k_{\varepsilon i}) \bmod d_A/2 \qquad (4.9.3)$$

will never hold.

Now, observe that, for $k_{\varepsilon j} = k_{\varepsilon i}$, i.e., $\varepsilon_i = \varepsilon_j$, Congruence (4.9.3) becomes:

$$\tau \equiv 0 \mod d_A/2 \qquad (4.9.3)$$

or, recalling that $\tau \geq 1$, $\tau = h\ d_A/2$, where h is a positive integer. Hence, for the code be 2 - detecting, necessarily:

$$\max \tau = n - 1 < d_A/2 = (A - 1)/2 \quad (4.9.4)$$

In the complementary hypothesis where $k_{\varepsilon j} \neq k_{\varepsilon i}$, i.e., $\varepsilon_i \neq \varepsilon_j$, from Congruence (4.9.3) and (4.9.4) it follows that

$$\tau = |k_{\varepsilon j} - k_{\varepsilon i}|\ d_A/2$$

and, necessarily, for the code be 2 - detecting

$$\max \tau = n - 1 < \min_{\varepsilon i, \varepsilon j\,:\, \varepsilon i \neq \varepsilon j} |k_{\varepsilon j} - k_{\varepsilon i}|\ d_A/2$$

This complete the proof of the necessity.

Sufficiency. Suppose that the condition of the Theorem holds and, by contradiction, assume that the code is not 2 - detecting. Then, there will be two integers ε_i , ε_j in [1, b - 1] such that

$$\varepsilon_i\, b^{\tau} \equiv \pm\, \varepsilon_j \mod A$$

with $1 \leq \tau \leq n - 1$ or, equivalently, recalling the above notations

$$b^{k\varepsilon_i}\, b^{\tau} \equiv b^{k\varepsilon_j} \mod A \qquad (4.9.1)$$

$$b^{k\varepsilon_i}\, b^{\tau} \equiv - b^{k\varepsilon_j} \mod A \qquad (4.9.2)$$

In order to Congruence (4.9.1) or Congruence (4.9.2) be verified the congruence

$$\tau \equiv (k_{\varepsilon j} - k_{\varepsilon i}) \mod d_A$$

or congruence

$$\tau \equiv (k_{\varepsilon j} - k_{\varepsilon i}) + d_A/2 \mod d_A$$

must hold. However:

- if $k_{\varepsilon j} = k_{\varepsilon i}$, then $\tau = d_A$ or $\tau = d_A/2$
- if $k_{\varepsilon j} \neq k_{\varepsilon i}$, then $\tau = |k_{\varepsilon j} - k_{\varepsilon i}|\ d_A$ or $\tau = |k_{\varepsilon j} - k_{\varepsilon i}|\ d_A/2$

In both cases, a contradiction follows and the proof is completed.

References

[1] D. M. Mandelbaum, "Error correction in residue arithmetic", IEEE Trans. Comput., vol. C - 21, pp. 538 - 545, June 1972

[2] F. Barsi and P. Maestrini, "Error detection and correction by product codes in residue number systems", IEEE Trans. Comput., vol. C-23, pp. 915 - 924, Sept. 1974

[3] D.M. Mandelbaum, "On a class of arithmetic codes and a decoding algorithm", IEEE Trans. Inform. Theory, vol. IT - 22, pp. 85 - 88, Jan. 1976

[4] F. Barsi and P. Maestrini, "Improved decoding algorithm for arithmetic residue codes", IEEE Trans. Inform. Theory, vol. IT - 24, pp. 640 - 643, Sept. 1978

[5] A. Shiozaki, “Single Asymmetric Error-Correcting Cyclic AN Codes”, IEEE Trans. Comput., Vol. C-31, pp. 554 – 555, June 1982

[6] R. Shimada, Y. Ohkura, J. Aoe, “Nonbinary Arithmetic AN Codes Using Odd Radix Expressions”, IEEE Trans. Comput., vol. C-34, pp. 1050 – 1056, November 1985

[7] D. T. Brown, "Error detecting and error correcting binary codes for arithmetic operations", IRE Trans. Electron. Comput., vol. EC - 9 , pp 333 - 337, Sept. 1960

[8] T.R.N. Rao and A.K. Trehan, "Single error correcting non binary arithmetic codes", IEEE Trans. Inform. Theory, vol. IT - 16, pp. 604 - 608, Sept. 1970

[9] T.R.N. Rao, *Error Coding for Arithmetic Processors*, Academic Press, New York, 1974

[10] T.R.N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*, Prentice Hall Series in Computer Engineering, Englewood Cliffs, New Jersey, 1989

[11] W.W. Peterson and E.J. Weldon, Jr., *Error Correcting Codes*, Cambridge MIT Press, 1972

[12] J.L. Massey and O.N. Garcia, "Error correcting codes in computer arithmetic", in *Advances in Information System Sciences*, ed. J.L. Tow, vol. 4, pp. 273 - 326, Plenum Press, New York, 1971

[13] M. Goto and T. Fukumura, “Nonbinary AN Codes with Distance not less than five”, IEEE Trans. Inform. Theory, vol .IT-19, pp. 129 – 134, January 1973

[14] A.C.L. Chiang and I.S. Reed, “Arithmetic Norms and Bounds of the Arithmetic AN Codes”, IEEE Trans. Inform. Theory, vol. IT-16, pp.470 – 476, July 1970

[15] R.M. Roth and G. Seroussi, “Reduced – Redundancy Product Codes for Burst Error Correction”, IEEE Trans. Inform. Theory, vol. IT-44, pp.1395 – 1406, July 1998

[16] H. Xu and F. Takawira, “A new structure of single parity check product codes”, Proceedings of the IEEE AFRICON 2004, pp 67 – 70

[17] S.A. Miri and A.K. Khandani, “On structure and decoding of product codes”, Proceedings of ISIT 2000, Sorrento, Italy