Error Codes in Digital Data Communication Systems

Robert Hadley Cravens
Portland State University

ERROR CODES IN DIGITAL DATA COMMUNICATION SYSTEMS

by

ROBERT HADLEY CRAVENS

A thesis submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE
in
APPLIED SCIENCE

Portland State University
1977

AN ABSTRACT OF THE THESIS OF Robert Hadley Cravens for the Master of Science in Applied Science presented July 29, 1977.

Title: Error Codes in Digital Data Communication Systems

APPROVED BY MEMBERS OF THE THESIS COMMITTEE:

[redacted]

Jack C. Riley, Chairman

[redacted]

Pah I. Chen

[redacted]

Ralph D. Greiling

[redacted]

Jerry Murphy

Today's digital communication systems perform data transfers at the rate of millions of bits per minute, with data errors in the order of 1/6th error per day. This magnitude of errorless communication is now possible because of sophisticated error correcting codes. Many types of error codes are employed today in three distinct areas of digital data communication: human to computer; data source to computer; computer to computer; and intra-computer; we are concerned here with intra-computer communication.

This research is primarily a mathematical study of error codes in general to explore the possibilities of each major type for the purpose of implementation in real systems. The author was inspired toward this goal by several people and self feelings. The first, was a definite

affinity toward orderliness and the logical sequence of formal mathematics. Secondly, the thrusting of being assigned to a work project where computer maintenance and where all types of errors became important. And, finally an advisor who believes in "practical things".

The original portion of this endeavor is to be found in the conclusions drawn from each group of mathematical facts disclosed in the research. The particular bend of the author toward the cost/reliability/ efficiency of the system was not the intent of the theoretical mathematicians who did the majority of the work quoted herein. The author's contribution was to draw these ideas and works together and to form the conclusions based upon his experience and training as an Engineer.

The primary conclusion is that multi-residue systematic codes appear to be the best choice for implementation of all around error correction and general hardware configurations. This conclusion is within the constraints that were laid down in the introduction of the research; 1) to not increase the cost of hardware, 2) to maintain or improve the system reliability, and 3) to maintain or increase the processing speed.

ACKNOWLEDGMENTS

The author acknowledges his appreciation to Professor Jack Riley for his invaluable help in sorting out the bureaucratic entanglements, and his patience and collaboration in the preparation of this thesis.

Also, the author wishes to express his gratitude to his wife, Maureena and family, Danielle and Kjell for their understanding of the disruption of their lives.

Finally, and by no means least, the author needs to express his thanks to Donna, the Super-Secretary, who typed, reviewed, critiqued and "got-it-right", this thesis; partly during her vacation, thank you!

TO THE OFFICE OF GRADUATE STUDIES AND RESEARCH:

The members of the Committee approve the thesis of Robert Hadley Cravens presented July 29, 1977.

[redacted]

Jack C. Riley, Chairman

[redacted]

Pah I. Chen

[redacted]

Ralph D. Greiling

[redacted]

Jerry Murphy

APPROVED:

[redacted]

Fred M. Young, Head
Department of Engineering and Applied Science

[redacted]

Stanley E. Rauch, Dean
Graduate Studies and Research

TABLE OF CONTENTS

PAGE

ACKNOWLEDGMENTS iii

LIST OF TABLES vi

LIST OF FIGURES vii

CHAPTER

INTRODUCTION 1

I ARITHMETIC CODES 3

Definitions 3

Classification and Definitions of Arithmetic Codes in a Finite Ring 7

Separate and Non-separate codes 8

Systematic and Non-systematic Codes 9

Distance and Error Correction Capabilities 10

Implementation of Separate and Non-separate Codes 13

Computer Architecture for Separate and Non-separate Codes 13

II CYCLIC CODES 18

Definition of Cyclic Codes 18

Large Distance Cyclic Codes 20

Mandelbaum-Barrows Codes 20

Generalized Large Distance Codes 22

III THE CONSTRUCTION OF MULTI-RESIDUE CODES 24

The Working Hypothesis 24

PAGE

Correspondence Between Separate and Non-Separate Codes........ 25

Errors in the Adder........ 25

Errors in the Checkers........ 27

Errors in the Adder or Checkers........ 28

Applications and Some Examples........ 28

Bi-Residue Codes........ 29

Other Multi-Residue Codes........ 31

IV ARITHMETIC ERROR CORRECTIONS........ 33

Simultaneous Failure of the Adder and the Checker........ 33

Corrections of Single Errors........ 34

Introduction........ 34

Solution........ 35

V SUMMARY AND CONCLUSIONS........ 38

A SELECTED BIBLIOGRAPHY........ 40

APPENDIX A........ 43

APPENDIX B........ 45

APPENDIX C........ 49

LIST OF TABLES

TABLE		PAGE
I	Mandelbaum-Barrows Large Distance Codes.................	21

LIST OF FIGURES

FIGURE PAGE

1 Implementation of AN Codes.............................. 14

2 Implementation of Multi-Residue Codes.................... 15

3 Implementation of Non-Separate Systematic Codes......... 16

4 Basic System.. 35

5 System for Correction of Single Errors.................. 36

INTRODUCTION

The research reported here concerns the theory and the application of error generation codes as used for error correction in the arithmetic operations of digital computers. These codes of varying types and designs are used by small microprocessors up through the huge data processing machines used for number manipulations. The errors may be due to transient or permanent component failures, to malfunctions due to electrical noise, or to both. The relevance of this topic is justified by the increased reliability requirements of modern arithmetic and logical processors, due to the state-of-the arts' growth in size, speed and the interest for Real Time operation. This endeavor will only cover binary processes' and therefore, only binary codes.

The greatest majority of today's digital code communication is intra-computer. It quite probably approaches 95% of all digital communications. Today's trend is to move toward large scale digital data communications to interface directly with computers and for long range telecommunications. Digital communications will offer many advantages as long as it is immune to errors or has the ability to correct those errors which do occur.

The recent history of methods to improve the reliability of digital processors shows three basic trends: 1) Computer Users have increased the reliability of the given system at the component level by using the best components and providing multi-level redundancy of component subsystems to be switched in place of a faulty one, 2) reliability of the processor is improved by providing alternate units, either to be switched in place of a faulty one, or to be operated simultaneously with some majority decision elements operating from the outputs of the duplicated units, or finally 3) the reliability is achieved by redundant information processing so that certain kinds of faults can be detected and/or corrected by non-complex circuitry. In the case of correction, the circuitry uses the redundant information to produce a result in which the effects of the malfunction are eliminated.

The goal of any error code generation scheme for arithmetic and/or logical operations is to provide correct results in the presence of component(s) failures without unduly increasing the cost, decreasing the system reliability, or slowing down its processing speed.

CHAPTER I

ARITHMETIC CODES

In this chapter the arithmetic codes are defined and classified, the fundamental definitions of arithmetic weight and distance are given, and the relationship between binary arithmetic distance and the error-correcting capabilities of the code are discussed. The treatment of arithmetic weight and distance differs from the orthodox approach shown by Massey [1] in that the binary operations are taken among the elements of a finite ring. This leads to the concepts of modular binary arithmetic weight and distance. Finally the most common digital architectures for the implementation of arithmetic codes are considered and evaluated.

DEFINITIONS

Arithmetic Weights and Distance

The following definition of binary arithmetic weight is from Peterson [2]. Let Z be an infinite ring of integers.

Definition 1.1 Arithmetic Weight:

Let N be an element of Z. The binary arithmetic weight of N, denoted BAW(N), is the minimum possible number of non-zero terms in the representation

$$N = a_n 2^n + a_{n-1} 2^{n-1} + \dots + a_o \qquad (1.1)$$

where n may be as large as needed, and a_i is 1,0, or -1 for i = 0, 1,

... n.

EXAMPLES: $N_1 = 31 = 2^5-1$, $BAW(N_1) = 2$

$N_2 = 11 = 2^4-2^2-1$, $BAW(N_2) = 3$

It should be noted that the minimum weight representation of an integer, i.e., the representation with the minimum number of non-zero terms in the expression (1.1) is not unique. For example,

$$N = 11 = 2^4-2^2-1 = 2^3 + 2^2-1.$$

Reitwiesner [3] has shown that if an integer is given in such form that the product (a_i) $(a_{i-1}) = 0$ in (1.1) for all $i = 1, \ldots, n$ then it is expressed in minimal weight form. An algorithm to determine by inspection the arithmetic weight of an integer expressed in binary form is given in APPENDIX A.

Definition 1.2 Arithmetic Distance:

Given N_1 and N_2 in the infinite ring of integers Z, the binary arithmetic distance between N_1 and N_2 denoted $BAD(N_1N_2)$ is given by

$$BAW(N_1-N_2) = BAW(N_2-N_1) \tag{1.2}$$

which also comes from Peterson [2]. If a number N_1 is transmitted and $N_1 \neq N_2$ is d, then a d-fold arithmetic error (E) is said to have occurred. In other words, let $N_2 = N_1 + E$ and then $BAW(E) = d$. The minimum d occurs when one bit position fails but when the failure is in one of the cells of a register or accumulator several consecutive bit positions will be affected due to propagation. The definitions of arithmetic weight and distance treat errors only as single errors.

Definition 1.3 n-Tuples:

Consider n-tuples, like

$$X = [X_o, X_1, \ldots, X_{n-1}] \tag{1.3}$$

where each component X_i is an element of a ring of integers modulo m_i for $i = 0,1,\ldots,n-1$. Garner [4] shows for example, $X = [0,-3,5]$ is a triple where the components 0,-3,5 are elements of rings of integers modulo 30,5,6 respectively.

From this point of view we could then treat the n bit binary expansion of an n-tuple for which $m_i = 2$ for $i = 0,1,\ldots,n-1$. An example would be as follows, a binary sequence like 10110111 representing the integer 183 is equivalent to [1,0,1,1,0,1,1,1] where $m_i = 2$ for $i = 0,1,\ldots,7$. In what follows the integer (183) will be treated as a sequence [10110111] or as an n-tuple [1,0,1,1,0,1,1,1] as needed. The appropriate interpretation should be clear from the context in which it is used.

Comparison and operations between n-tuples will be defined only when they have the same number of components and the corresponding components belong to the same ring of integers modulo m_i for $i = 0,1,\ldots,n-1$.

Definition 1.4 Congruency:

An integer N is said to be congruent to an integer N_2 modulo q_i, denoted $N_1 \equiv N_2 \bmod q_i$, if

$$N_1 = kq_i + N_2 \tag{1.4}$$

for some integer k. Garner [4] again, shows how to prove congruency.

Two n-tuples X and Y are equal if all their corresponding components are congruent, i.e., if $x_i = y_i$ for $i = 0,1,\ldots,n-1$.

Operations between n-tuples are defined component wise, i.e., as operations in the ring of integers modulo m_i for the corresponding components. Further two operations denoted $\oplus$ and $\otimes$, are meaningful between n-tuples, they correspond exactly to the operations of addition and multiplication in the rings of integers of the components.

Given an integer n, the least non-negative integer congruent to N modulo m_i is represented by $|N|_{m_i}$ and is usually called the residue of N modulo m_i. Then we can write, using (1.2)

$$X \oplus Y = \left[|x_0 + y_0|_{m_0}, \ldots, |x_{n-1} + y_{n-1}|_{m_{n-1}} \right]$$

where + is used for ordinary addition.

The concepts of BAW and BAD given before differ from the concepts of Hamming Weight (HW) and Hamming Distance (HD) which are also very useful in communication codes. From Hamming [5], then the following definitions.

Definition 1.5 Hamming Weight:

Given an n-tuple $X = [x_0, x_1, \ldots, x_{n-1}]$ defined from (1.3) its Hamming Weight, represented HW(X), is the number of non-zero components in

$$X = \left[|x_0|_{m_0}, |x_1|_{m_1}, \ldots, |x_{n-1}|_{m_{n-1}} \right] \tag{1.5}$$

From the previous definitions and the interpretation of the binary expansion of any integer N as an n-tuple we deduce, with the insight of Rao [6], that

$$HW(N) = BAW(N).$$

Definition 1.6 Hamming Distance:

The Hamming Distance between two n-tuples X and Y is

$$HD(X-Y) = HW[(x_o - y_o),(x_1 - y_1),\ldots,(x_{n-1} - y_{n-1})] \qquad (1.6)$$

There is a margin of error in finite ring arithmetic as used in generating error codes but it can be dealt with by using certain constraints. For deeper mathematical involvement the reader should adjourn to APPENDIX B.

CLASSIFICATIONS AND DEFINITIONS OF ARITHMETIC CODES IN A FINITE RING

An error code can be used to detect and/or correct errors in basic arithmetic and arithmetic related operations. All error codes depend on some form of redundancy; only error codes which are finite number systems are considered because the application to computers and other digital communication schemes is finite. Any useful arithmetic code would have to at least check addition since the basic computer operation is addition: related operations needing error correction are shift, rotation, and complimentation. Although this paper is concerned only with the correction of addition errors, work has also been done on multiplication errors.

Garner [4] has shown that when the error code is an ideal in a finite ring of integers, it is possible to detect and/or correct errors in multiplication as well as in addition. Binary radix and diminished radix complement codes form additive groups modulo $m_o = 2$ or $m_o = 2^n-1$ respectively, and Garner [7] has also shown how to model these codes as rings for multiplication.

Separate and Non-Separate Codes

We can classify the error codes into two broad groups according to how the arithmetic is performed on the n bits representing the code words. If exactly the same arithmetic rules are applied to all n bits with the possible exception of an additive or multiplicative correction after the operation is performed, then we have a non-separate code. If however, different, independent, arithmetic is specified for the k information carrying bits and for the redundant (m-k) remaining bits, such that there is no transfer of carries (or overflows) from one operation to the other, then we have a separate code.

A general type of non-separate error code is the AN+B code originally considered by Diamond [8] and Brown [9]. In these codes, the information represented by a number N is multiplied by a constant A; the additive constant B is sometimes used to make the complimentation of the coded words simpler, but in the following cases we will set B = 0. There is also a requirement that A must have at least some non-trivial factor relatively prime to the base b of the system in which the error code is represented. This is to circumvent the single errors going unnoticed. A theorem credited to Fisher by Szabo and Tanaka [10] shows that a non-separate code, as defined above, must be of the type AN if the code is to be preserved under addition and subtraction. Garner [4], more precisely, gives the necessary and sufficient conditions for the existence of a non-separate code. However, existence alone does not give any information of the error correcting capabilities of the code. A very interesting subclass of non-separate codes called "Systematic

Codes" was discovered by Henderson [11]; they will be considered in detail in the next section.

A general type of separate code is the multi-residue code in which the information representation by the number x in the ring of integers modulo m_o is coded as a (k + 1)-tuple X, and the moduli m_i for $0 < i < k$ are pairwise relative prime integers. This is actually a generalization of the bi-residue code as defined by Rao [6]. The arithmetic of multi-residue coded words is performed as in Definition 1.2. The independent arithmetic of the residues allows a separation between the operation being monitored and the error detecting and error correcting processes. Peterson [12] has shown that modular arithmetic is the only way of separately checking addition with fewer digits than are used in the adder. Garner [4] extended the same result to include multiplication.

Systematic and Non-Systematic Codes

The use of the word systematic comes to arithmetic coding theory from codes used in communication. Unfortunately, it has been used by Szabo and Tanaka [10] as a synonym of separate, which is not correct if we extend its meaning from that accepted for communication codes. Garner [4] clearly has pointed out that in separate codes there is no interaction between the arithmetic of the information part and the redundant checks; he also points out that it is possible to have some arithmetic interaction from the information part to the redundant checks and yet preserve intact the information portion. It is for these codes, in which the information bits always remain intact after coding, that the term systematic is reserved.

Definition 1.7 Systematic Code:

Let $(a_1,a_2,\ldots,a_k)$ be a sequence of information symbols which when coded, form a code word $(b_1,b_2,\ldots,b_n)$ for $n>k$. If we can identify each symbol a_i in the information sequence with a distinct b_j in the code word, and the correspondence $i \rightarrow j$ in positions is fixed for all code words, then the code words form a systematic code. And conversely, any code not satisfying Definition 1.6 is therefore non-systematic.

Now it can be stated that systematicity cannot exist if there is the possibility of carries from the check positions flowing into the information positions. This is a very important consideration in design of the correction circuits for typical adders and checkers.

From this definition and Von Neuman's [13] work it can be shown that all separate codes are trivially systematic. What is more interesting however, is the fact that there exists a class of non-separate codes which are systematic; these codes are formed by a left concatenation of the check symbols. Under the usual assumption that carries flow to the left, except perhaps out of the most significant bit position, we see that the check symbols must be to the left of the information symbols to preserve systematicity. Systematic codes are particularly attractive because the information symbols are readily available without processing, except where error corrections are to be made on them.

Distance and Error Correction Capabilities

Peterson [2] remarked that the arithmetic error code must be an ideal in Z_{m_o}; furthermore, he says that it must be a principal ideal since Z_{m_o} is a ring which contains only ideals generated by a single element, i.e., principal ideals. The following is from and by Peterson [2]:

LEMMA I: Z_{m_o} is a principal ideal ring (P.I.R.), i.e., a ring in which any proper ideal is generated by a single element of the ring.

PROOF: Assume that there is a proper ideal in Z_{m_o}, generated by two elements, say p and q of Z_{m_o}. If GCD(p,q) = d then there would exist integers m and n such that

$$mp + nq = d$$

and where $mp = \underbrace{p+p+\ldots+p}_{m \text{ times}}$ and $nq = \underbrace{q+q+\ldots+q}_{n \text{ times}}$

and then d would be in the ideal; but this then, would be the ideal generated by d, a single element, contrary to the hypothesis. If d = 1 then the ideal is not proper. We could easily extend the argument to any finite number of elements. Therefore, the only proper ideals in Z_m are principal ideals.

This relationship is visualized in AN codes in which A is the generator of the ideal in the ring of integers modulo m_o. This implies that A divides m_o and then, given any two words AN_1 and AN_2 in the code

$$|AN_1 - AN_2|_{m_o} = AN_3$$

is also in the code. The minimum MBAD (Modular Binary Arithmetic Distance) between two code words is equal to the minimum MBAW of the code word AN_3.

If an erroneous result, as in Definition B.1 (see APPENDIX B), occurs due to one or more failures, it can be said that MBAW(E) = t errors have occurred, or that E is a pattern of t errors. In general in parallel adders, the independence of the paths allows for the correlation of the number of errors so obtained with at least the same number of failures of electronic components in the adder, as software errors.

Massey [1] proved that an AN (Arithmetic Product) code could correct all patterns E of t or fewer errors, if and only if, the minimum weight of all code words is equal to or larger than two times the number of errors plus one, or (2t + 1). The following notation is by Peterson and Rakin [12] who did some of the first work in error codes.

If A is a generator of an AN code in base b representation, then $M_b(A,d)$ is the smallest non-negative integer whose product with A has a BAW less than d. And if b = 2 (binary) the subscript will be dropped; the integer is denoted by M(A,d). Now Brown [9] added two theorems and a third theorem was proven independently by Henderson [11] and Peterson [2] which when used in conjunction brings us to the codes being used today in the more advanced correction schemes. These theorems basically state that: 1) any positive odd integer A generates a code which detects all error patterns for large values of information, 2) given the conditions of Theorem 1 then M(A,3) is the least positive integer which results from $\frac{2^k \pm 1}{A}$ for some positive integer k, 3) if A > 1 is an odd prime and if all the non-zero elements of the field of integers modulo A may be generated by 2^i or by $(-2)^i$ for suitable integer values of i.

The use of separate codes for error detection has been known for a long time. The procedure of "casting out nines" is actually an error detecting residue code for the decimal system. It is generally known that any odd m_1 will give a single residue code capable of single error detection in the arithmetic of a binary number system. If m_1 is of the form $2^x - 1$ there are simplifications in the computation of the residues. The choice $m_1 = 3$ is the most popular.

The errors in separate code arithmetic may originate in either the adder or the checker. The use of single residue codes makes error correction impossible because the determination of the error location is not possible. The fault might be in the checker but would erroneously indicate a fault in the adder. However, Rao [6], developed a means of single error correction using bi-residue codes that will be introduced in Chapter II.

IMPLEMENTATION OF SEPARATE AND NON-SEPARATE CODES

The selection of a good arithmetic code means one which can detect and correct the error patterns which are most likely to occur with as simple a detection and correction implementation as feasible. If the implementation of the code requires an excessive amount of additional hardware, it is conceivable that the overall reliability of the coded system will be lower and the cost higher than that of the original non-redundant system.

Computer Architecture for Separate and Non-Separate Codes

The implementation of AN codes requires: 1) an encoder which forms the product of the constant A and the information N; 2) an arithmetic unit which performs the addition of two encoded words AN_1 and AN_2; 3) a decoder which given the result $A(N_1 + N_2) + E$ finds $|A(N_1 + N_2)+E|_A$ and implements a correction if $|E|_A$ is non-zero and is a correctable error, or perhaps sets some error procedure for having detected the error. This procedure is shown in block form in Figure 1.

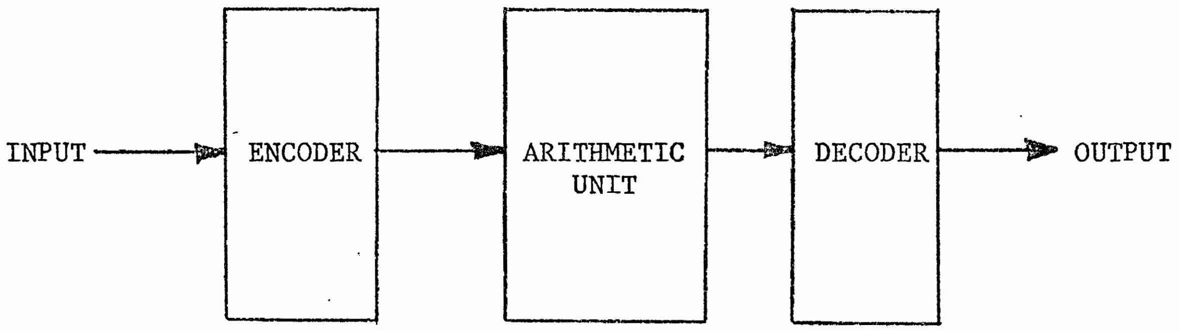

Figure 1. Implementation of AN codes

The system of Figure 1 may be capable of controlling both transient and permanent faults in a computer system, but the slower speed and the added cost over the original system is not an equitable trade-off in view of the limited error correction provided. The relatively small range of values allowed for N especially would make the cost and time factors when considering the word size of todays' large computer systems. Several previous researchers have suggested utilizing the arithmetic unit to encode and decode. This approach suffers from two major drawbacks: 1) it still will show transient failures but does not retain the ability to accurately handle permanent failures, and 2) it slows down the system operation further. The drawbacks of non-separate codes far outweigh the advantages.

A second possibility for error control using a multi-residue code is shown in Figure 2.

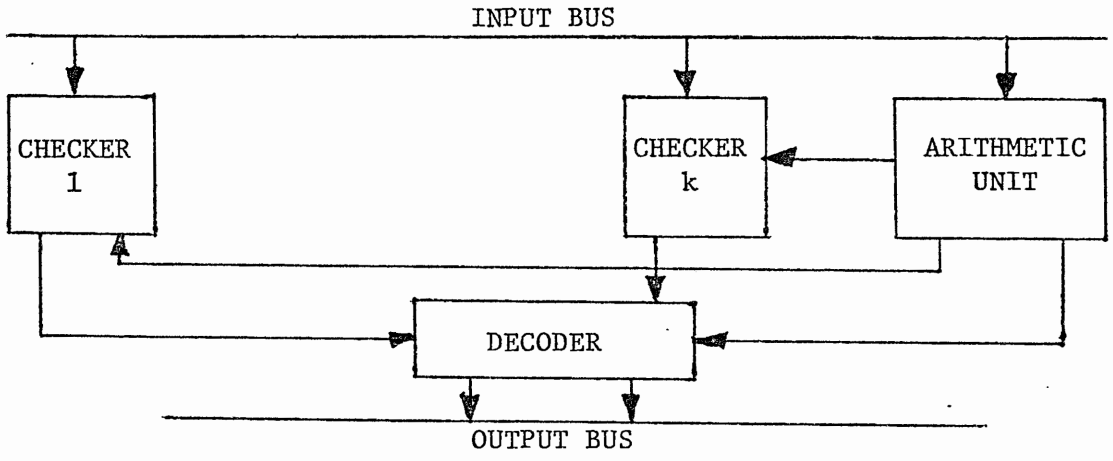

Figure 2. Implementation of multi-residue codes

Here, addition is also slowed since we must wait for the result to be formed and then decide and select a correction if needed. The arithmetic unit or adder as it is called in the use of multi-residue codes, though it may have other arithmetic functions, contains the m_o component of the coded word. The other k units which perform arithmetic modulo m_i for $i = 1,2,...,k$ are called the modulo m_i checkers or checkers for their respective value of i. Still this code and hardware does not, under test, give us the best values possible.

The final case for non-separate codes is helped by Garner's [4] investigation of a method of forming the residue of the sum by starting with the least significant bits, at a speed very close to the velocity of carry propagation. For preselected moduli this scheme could improve the correction time for the non-separate configuration. This approach and a fast table-look-up procedure for the correction corresponding to each syndrome could reduce even further the loss of processing time for correction.

It should be pointed out that the non-separate systematic codes are, in fact, partially modular, since there is only one-directional interaction, from the information bits to the check bits. The Figure 3 block diagram shows one possible design for the implementation of these "hybrid" codes.

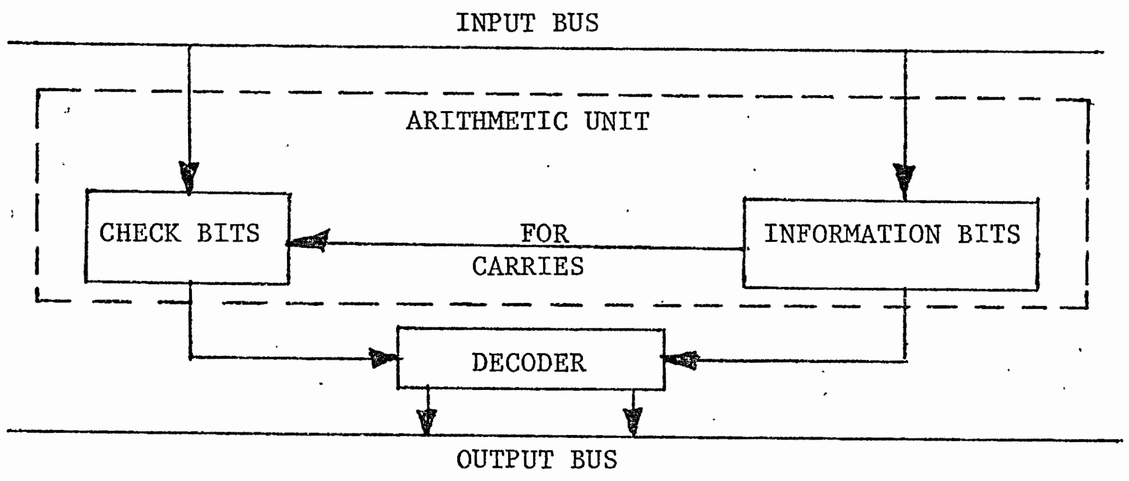

Figure 3. Implementation of non-separate systematic codes

The separate codes as compared to the non-separate AN codes have several advantages. Rao [6],[14] in his articles on computer processors and bi-residue codes pointed out these four: 1) the arithmetic unit and the checkers operate independently in the sense that faults in any one unit will not normally contaminate the others; 2) error control of both transient and permanent faults is possible; 3) the range of values of the information N under error control is now $0 \leqslant N < AM$ much larger than $0 \leqslant N < M$ for nonseparate codes; and 4) implementation is relatively easy, especially so if the arithmetic unit uses one's complement system (i.e., $m_o = 2^n-1$), and the moduli m_i of the checkers are of the type 2^x-1 such that x divides n exactly.

One distinct advantage of separate codes is that in general the theory relating their distance properties and error correcting capabilities has been well developed. This along with the above mentioned advantages and lower time requirements in terms of CPU time make them an ideal choice for correcting codes.

CHAPTER II

CYCLIC CODES

In this Chapter a class of codes called cyclic are defined and studied. The special advantages are noted and discussed; the primary importance being that this class of codes is capable of multiple error correction. The results of the error correcting capabilities of these cyclic codes is important because more practical, separate codes have been constructed with properties which are analogous to them. And finally a discussion about the theory of codes capable of multiple error corrections (large distance codes) completes the Chapter.

DEFINITION OF CYCLIC CODES

The consideration here is the finite AN codes useful for binary arithmetic operations. If the arithmetic registers are of the size n bits (or of n gates), then each AN code word is represented as a binary n-tuple and N will have a range $0 \leqslant N < M$ where $AM < 2^n$. Then the code is said to be of length n. A cyclic AN code can now be defined as follows:

Definition 2.1 An AN code of length n is said to be cyclic if and only if for any code word $X = [X_o, X_1, \ldots, X_{n-1}]$, the word $Y = [X_1, X_2, \ldots, X_{n-1}, X_o]$ obtained by rotating the bits of X to the left once, is also a code word.

This definition and the theorem that follows are quite analogous to the cyclic codes for communication theory, for examples see Peterson [2], page 137. Peterson [2] also puts forth this theorem with the partial objective of bridging the "algebraic foundations" of the arithmetic codes and the communication codes. For this reason, and the convenience in arithmetic operations, he used this restrictive definition rather than to define all finite AN codes that are closed under addition (which form a cyclic subgroup in the additive group of integers mod AM) as cyclic.

The necessary and sufficient conditions for a code to be cyclic are given in this theorem.

Theorem 2.1 An AN code of length n is cyclic if and only if A generates an ideal in R, the ring of integers modulo 2^n-1.

Proof: Let the AN code by cyclic. That is, if the word $X = [X_o, X_1, \ldots, X_{n-1}] = AK$ for some K, $0 \leqslant K < \frac{2^n-1}{A}$, then $Y = [Y_1, Y_2, \ldots, Y_{n-1}, Y_o]$ is also a code word and hence a multiple of A.

Let

$$X = \sum_{i=o}^{n-1} x_i 2^i = AK$$

then

$$Y = \sum_{i=o}^{n-1} x_i 2^{i+1} + x_{n-1} = \begin{cases} 2AK & \text{if } x_{n-1} = o \\ 2AK - (2^n-1) & \text{if } x_{n-1} = 1 \end{cases}$$

Since Y is a code word for either case, and this can be true only if A divides 2^n-1, then, it follows from elementary theory of rings, A generates an ideal in R.

Conversely, if A generates an ideal in R, then A divides 2^n-1. Then for all X = AK, the cyclic shift to the left of X gives us a Y such that

$$Y = \begin{cases} 2AK & \text{for } x_{n-1} = o \\ 2AK - (2^n-1) & \text{for } x_{n-1} = 1 \end{cases}$$

In any case A divides Y and hence Y is a code word, thus proving that AN is cyclic.

Cyclic codes lend themselves very well to both rotation and complementation operations and are often involved in those arithmetic operations. From the definition of cyclic codes it is clear that rotation (both left and right) is a closed operation when performed on AN cyclic code words. Since this operation may be considered as a transmission operation, the number of errors which may be detected, or corrected under rotation, is determined by the minimum Hamming Weight (HW) of the code words, which is no smaller than the minimum Modular Binary Arithmetic Weight (MBAW) of the code words. Complementation of cyclic AN codes is also closed and therefore its complement is also a cyclic code word. The detectable and correctable errors in complementation are specified by the minimum Hamming Weight.

LARGE DISTANCE CYCLIC CODES

The codes that have been discussed so far have been single error correcting, i.e., of distance 3. The term "large distance" is not specific, so for this remaining discussion a distance of 4 or greater will be termed large.

Mandelbaum-Barrows Codes

Mandelbaum [15], and independently, Barrows [16] have analyzed non-separate codes with A of the form

$$A = \frac{2^{e(B)}-1}{B} \tag{2.2}$$

where B is a prime for which e(B) = (B-1). Under these circumstances, Barrows found that

$$M(A, \left[\frac{B+1}{3}\right]) = B \tag{2.3}$$

where the integer part of $\frac{B+1}{3}$, denoted by $\left[\frac{B+1}{3}\right]$, is the minimum weight of all the code words in the ring of integers modulo $m_o = 2^{B-1}-1$. Since then Hong [17] and Chang, with Tsau-Wu [18], reached the same goal from a different approach.

These codes are cyclic since they satisfy the condition of Theorem 2.1. Some examples of these codes are shown in TABLE I: giving, the generation A, in smallest prime factors; the prime B; and d min, the minimum distance in the ring of integers modulo $2^{b-1}-1$.

TABLE I

MANDELBAUM-BARROWS LARGE DISTANCE CODES

A	B	d_{min}
(3)(31)	11	4
(3)(3)(5)(7)	13	4
(3)(3)(7)(73)	19	6
(3)(5)(43)(113)(127)	29	10

The range for these AN codes is exactly 0 < N < B while they offer correction of $\left[\frac{B+1}{3}\right]$ errors is (B-1) bits. However, there is a draw-back on this type of correction code, the usual number of information bits used in general purpose computers is 30 to 40 bits and therefore the amount of redundancy required is prohibitive. The number of redundant bits is given approximately by Rao [19] as

$$\log_2 A = B - 1 - \log_2 B \tag{2.4}$$

It is now shown clearly that despite their large distance properties these non-separate codes are impractical because of their limited range and high level of required redundancy.

GENERALIZED LARGE DISTANCE CODES

Generalized large distance codes can be represented by

$$M_o = (2^n-1) = \prod_{i=1}^{m} p_i\alpha_i \tag{2.5}$$

For a given n let the prime factorization of 2^n-1 be as shown above in (2.5). The number of proper ideals in the ring of integers modulo m_o is equal to the number of proper divisors of m_o. All of these ideals are cyclic codes. If B is one of these divisors, of $m_o = 2^n-1$ and if its order of 2 is a proper divisor of n, we will find that the code obtained is a repetition of $\frac{n}{e(B)}$ times the basic binary patterns of a code of length e(B). For example: let $m_o = 2^8-1 = 3 \times 5 \times 17$ and $B = 5$. Then $e(B) = B-1 = 4$ and the generator if $A = \frac{2^8-1}{5} = 51$. Therefore the code words are (0,51,102,153,204). The non-zero code words have the following binary expansions

```
00110011
01100110
10011001
11001100
```

in which each word repeats the pattern in groups of four bits. This is very important because codes with repeating patterns are trivial in the sense that the increase in distance is obtained by replication. This generalization that B is prime allows us to consider a generator of the form (2.2) with $e(B) \neq B-1$, then we obtain codes with less redundancy than the Mandelbaum-Barrows codes, which therefore are more desirable codes in the light of cost and equipment.

CHAPTER III

THE CONSTRUCTION OF MULTI-RESIDUE CODES

In this Chapter the fundamental results which allow the construction of multi-residue codes corresponding to AN codes are given in such a way that under some natural restrictions their error correcting capabilities are alike. As discussed in Chapter I, the computer organizations necessary for separate and non-separate codes are quite different. It is, therefore, to be expected that the error-correcting capabilities of corresponding codes have to be considered with regard to their different organization. This is accomplished by establishing a working hypothesis on the failures and where they occur at the beginning of this Chapter. The two categories of errors are analyzed separately and then combined under the working hypothesis. In conclusion, some examples are given and the advantages of the separate codes obtained are enumerated upon.

THE WORKING HYPOTHESIS

In non-separate, non systematic arithmetic codes it is impossible to distinguish the information bits from the check bits, and therefore, it is meaningless to consider whether the error occurs in the information bits or in the redundant bits.

This is not so in the case of systematic codes. In the case of non-separate systematic codes the capabilities of the code are not a function of whether the possible error is located in the information part or in the check part. But, in separate codes, it is of fundamental importance whether the error occurs in the information part or in the redundant checkers.

As before, the circuit handling the first component of the multi-residue coded word will be called the adder, although it also could have other arithmetic functions. The adder will perform modulo m_o arithmetic. The arithmetic of the k redundant components of the multi-residue word is modulo m_i arithmetic, for $i = 1,2,\ldots,k$. This arithmetic is carried out in independent units which we will designate as the checkers.

In the arithmetic of separate codes three categories of errors can be distinguished: 1) errors in the adder and 2) errors in one or more of the checkers and 3) errors in both categories one and two simultaneously.

The following hypothesis is valid for the next section of this discussion: errors may occur either in the adder or in the checkers, but errors in both categories will not occur simultaneously.

CORRESPONDENCE BETWEEN SEPARATE AND NON-SEPARATE CODES

The first consideration will be the relationship between errors in the arithmetic modulo AM of non-separate codes and errors of the first category of separate codes in which m_o = AM and the m_i are pairwise relative prime factors of $A = \prod_{i-1}^{k} m_i$. The second consideration will be the possibility of distinguishing between the two kinds of errors and of correcting errors in the second category.

Errors in the Adder

There is a complete correspondence between errors in a non-separate AN code and errors of category 1 of a multi-residue code as specified by Peterson [12] as follows:

LEMMA 3.1 Let $A = \prod_{i=1}^{k} m_i$, where the m are pairwise relatively prime integers. Then given a non-separate (AN) code capable of correction all E in U(AM,d) there exists a separate multi-residue code $[x_1, x_2, \ldots, x_k]$ for a x in the ring of integers modulo m_o = AM and the m_i as given, which has a distinct syndrome corresponding to each E in $U(M_o, d)$ which may occur in the adder. Conversely if we are given a multiple residue code with distinct syndromes corresponding to each E in $U(m_o, d)$ in the adder, then the corresponding AN code has the same error-correcting capabilities.

PROOF: Let $s = |E|_A$ and $s' = |E'|_A$ be the syndromes corresponding to any E and E' in $U(m_o, d)$.

Given $\qquad |E|_A \neq |E'|_A \qquad$ (3.1)

we will have $\qquad |E|_{m_i} \neq |E'|_{m_i} \qquad$ (3.2)

for some i, for $1 \leqslant i \leqslant k$, because otherwise if

$$|E|_{m_i} = |E'|_{m_i}$$

for all i where $1 \leqslant i \leqslant k$, this would imply that there exists a positive integer k such that

$$(E - E') = K \langle m_1, m_2, \ldots, m_k \rangle \tag{3.3}$$

but since the m_i are pairwise relative prime their least common multiple equals their product

$$\langle M_1, m_2, \ldots, m_k \rangle = \prod_{i=1}^{k} m_i = A \tag{3.4}$$

and from (3.3) $\qquad |E - E'| = KA$

which contradicts our hypothesis (3.1). But then, since (3.2) is true, we have the syndromes

$$(|E|_{m_1},\ldots,|E|_{m_i}) \neq (|E'|_{m_1},\ldots,|E'|_{m_k}) \qquad (3.5)$$

This proves the first segment of the LEMMA 3.1, now conversely, let (3.5) hold, then (3.2) is true and (3.1) will hold since otherwise (3.4) would be true, which would imply (3.3) true, which means that

$$(|E|_{m_1},\ldots,|E|_{m_k}) = (|E'|_{m_1},\ldots,|E'|_{m_k})$$

which is contrary to the hypothesis but proves the second segment of the LEMMA 3.1.

Error In The Checker

Under the hypothesis stated at the beginning of the chapter, we can now predict the total number of checkers which may be corrected if they fail. A failure in the i^{th} checker is any malfunction which makes the corresponding component of the syndrome $s_i \neq |x - x_i|_{m_i}$, where x is the first component of the result (in the adder) being checked and x_i is the (i + 1) component of the same multi-residue codes result. Rao [15] shows how all possible errors can be predicted.

LEMMA 3.2 If s of the k checkers involved in the arithmetic of a multi-residue code with $m_o = \prod_{i=1}^{k} m_i$ (the m_i are pairwise relative prime) can independently detect all errors E in $U(m_o,d)$, then all simultaneous malfunctions in (s - 1) or fewer checkers are detectable.

PROOF: If a checker module m_i can detect all errors E in $U(m_o,d)$ then we are assured that for all such E the component $s_i = |E|_{m_i}$ is non-zero. Now we have (under the hypothesis that no errors of both categories will occur simultaneously) a criteria for localizing the error; since all errors E in the adder have at least s non-zero components in their corresponding syndromes, then any error with less than s non-zero components in their corresponding syndromes, then any error with less than s non-zero components must have been due to malfunction of the corresponding number of checkers.

Error In The Adder Or Checker

Now by summarizing the results of the previous LEMMAS and their philosophies the following is predictable for errors occurring in the adder and in the checker.

THEROEM 3.1 Let all errors E in U(AM,d) be correctable by an AN code such that $A = \prod_{i=1}^{k} m_i$ where the m_i are pairwise relatively prime. The corresponding multi-residue code with m_o = AM and the m_i as specifies for $1 \leqslant i \leqslant k$ is capable of correcting all E ε U(AM,d) in the adder, or all errors in (s-1) or fewer checkers if s of the checkers may each independently detect all E in U(m_o,d), under the assumption that if errors occur in the adder they do not occur in the checkers and vice-versa.

PROOF: LEMMA 3.1 guarantees distinctiveness of the syndromes of E for the multi-residue code, which suffices for their correctability.
Under the hypothesis that errors of both categories do not occur simultaneously, detection of errors in the checkers is guaranteed for (s-1) or fewer checkers in error by LEMMA 3.2.

APPLICATIONS AND SOME EXAMPLES

From what Chapter II covered about predicting the error-correcting properties of cyclic non-separate codes and THEOREM 3.1, we can predict most of the properties of a large class of separate codes, namely those formed with the pairwise relatively prime factors of the corresponding generator A.

Bi-Residue Codes

For single error correction the hypothesis given at the beginning of this Chapter is automatically fulfilled; a single error cannot affect the adder and the checkers at the same time. For correction, however, a minimum of two checkers are needed; because with only one checker, a failure in it could produce a syndrome corresponding to an error in the adder which has not actually occurred.

Two checkers, therefore, are the most economical configuration in number of modules for error-correction, and it is then reasonable to ask which are the most desirable moduli for single error correction. Rao [6] has investigated this problem and has found that a single error correcting bi-residue code may be constructed for integers in the ring $m_o = 2^n-1$ with $m_1 = 2^a-1$ and $m_2 = 2^b-1$ (m_1 and m_2 are not necessarily prime integers but only pairwise relatively prime) provided n is equal to the least common multiple of a and b, i.e.,

$$N = \langle a,b \rangle \tag{3.6}$$

The most efficient use of the redundant information is when GCD(a,b) = 1, and then $n = a \cdot b$. The choice of modulo of this type greatly simplifies the implementation of the code. Since LEMMA 3.1 guarantees also the same correcting capabilities of a non-separate code corresponding to a separate code as explained, we can affirm that AN codes with $A = (2^a-1)\cdot(2^b-1)$ for n as in (3.6) have

$$M(A,3) = \frac{2^n-1}{A} \tag{3.7}$$

These codes, however, are not of great practical interest as separate codes because of their limited range, but they may be easily implemented as bi-residue codes.

From Henderson [11] we know that given a certain $A = p_1 \cdot p_2$, we can find $M(A,3)$ from e_1 and e_2. This presents an immediate way of constructing single error correcting bi-residue codes. Given that the codes $p_1 n$ and $p_2 n$ are both single error detecting (and this is true for any odd p_1 and p_2), we can actually correct a single error E in the range $0 < E < AM \leqslant AM(A,d)$ in the adder or any error in one of the checkers.

When considering the practical implementation of these separate codes it is advantageous to have the prime moduli of the form 2^x-1, and also $AM = 2^n-1$. For example, let

$$p_1 = 7 \qquad p_2 = 31$$

$$e_1 = 3 \qquad e_2 = 5 \qquad d = (3,5) = 1 \text{ (odd)}$$

Then the lowest common multiple denoted $\langle , \rangle$ of $L(A)$ according to Henderson [11] is going to be $L(A) = \langle e_1, e_2 \rangle = 15$. Then

$$M = M(A,3) = \frac{2^n-1}{A} = \frac{2^{L(A)}-1}{(p_1 \cdot p_2)} = \frac{32768 - 1}{(7)(31)} = \frac{32767}{217} = 151$$

This code as a bi-residue code of the form $[N, |N|_y, |N|_{31}]$ will correct all single errors $\pm|2^i|_A$ in the adder for $0 \leqslant i < 15$ and for $0 \leqslant N < 32767$, and since the only requirement for single error detection is an odd modulus, we can also correct all errors in any one of the two checkers.

When p_1 and p_2 are not in the form of 2^n-1 the matter of use or implementation is open to question. Using Theorem 3.1, however, we can determine their error detecting and correcting capabilities. For example,

$[N,|N|_3,|N|_{29}]$ is also a bi-residue code capable of single error correction in a 28 bit adder or correction of any possible error in one of the two checkers. Notice $e_1(3) = 2$, $e_2(29) = 28$, $L(A) = 87$, and $0 \leqslant N \leqslant 268,435,455$.

Any of the Mandelbaum-Barrows codes, as those given in Table I, or any generalized large distance codes, may be used to form large distance bi-residue codes by suitably partitioning the generator A into two relatively prime factors.

It is important to notice that from these large distance non-separate codes, for which the range was so limited, we can now construct separate codes with a range many orders of magnitude greater. In the Mandelbaum-Barrows codes, for example, the range is given by the prime B; its corresponding multi-residue code has a range of $2^{B-1}-1$.

When the generator A is a composite number with more than two different prime factors, we have a certain flexibility in the way we may partition the factors to form the relatively prime moduli m_1 and m_2. The considerations are important in making this choice: 1) simplicity of implementation and 2) capability of error-correction. The former consideration is a problem presently open to question, unless the possibility of moduli of the type 2^x-1 exists, in which case this is the most desirable one. The latter consideration is not significant under our prevailing hypothesis, but will be taken up in the next chapter.

Other Multi-Residue Codes

Similarly, other separate codes may be constructed from non-separate codes where A has more than 2 distinct prime factors, using more than 2 checkers, one per each pairwise relatively prime factor. Under the hypothesis on the location of the errors, this would mean possibly greater capacity for error-correction. The option of how to group the factors of A remains open if A has more distinct prime factors than there are checkers since the only requirement is that the moduli be pairwise relative prime. The only other consideration is as before, the simplicity of implementation.

CHAPTER IV

ARITHMETIC ERROR CORRECTIONS

The fact of simultaneous failures of adders and checkers will be addressed in this chapter. Also, the implementation of bi-residue codes to correct errors in addition and other operations related to computer arithmetic will be considered. We are self restricted, of course, to failures which produce errors of the type $\pm 2^i$ in the result of the arithmetic operation. Single component failures originate errors of this type in the adder, in which the bits of the numbers to be added flow in independent parallel paths. This also includes single failures in the carry circuitry. Once the sign and magnitude are known, there are several possible strategies to correct the results in the accumulator. This chapter also evaluates the logical implementation of these different methods for correction.

SIMULTANEOUS FAILURE OF ADDERS AND CHECKERS

The assumption, made in Chapter III, that errors occur in either the adder or checkers, but not in both simultaneously, is based on the fact that the size of the adder is comparable with the size of all the checkers together, up to around three or four well chosen moduli. But, that assumption was only for the special cases of Chapter III and, if multiple errors are considered, they would distribute themselves among the adder and the checkers simultaneously.

The checker is assumed to have the same probability of failing as the adder, even though it is much smaller. This leads to a new concept of (k + 1) independent units with equal failure probability. Under this assumption a limit on the number of checkers which may fail and be corrected, while simultaneously having errors of a certain class in the adder, can be precisely determined. If Hamming [5] did his work correctly the number of correctable errors is given by

$$F = \frac{k - r(t)}{2} \tag{4.1}$$

where the right portion of the expression is the integer part of one-half of the minimum Hamming Weight of the syndromes associated with all $E \varepsilon U(m_o, 2t)$. Now, in a multi-residue code with k moduli, the maximum number of correctable checkers is $[\frac{k}{2}]$. This occurs when $r(t) = 0$, which occurs when the only error is found to be in the checker section of the correction scheme.

CORRECTIONS OF SINGLE ERRORS

Introduction

The basic system to be checked and corrected for single component failures is shown in Figure 4 in block diagram format. The transfer from memory is done in parallel through the memory buffer register (MBR) to the accumulator register designated as ACCUM. The accumulator does the addition by means of the special hardware connection and decoding of the ADD instruction, but on the figure it has been depicted as a separate block; the results are stored in the ACCUM after the addition is complete.

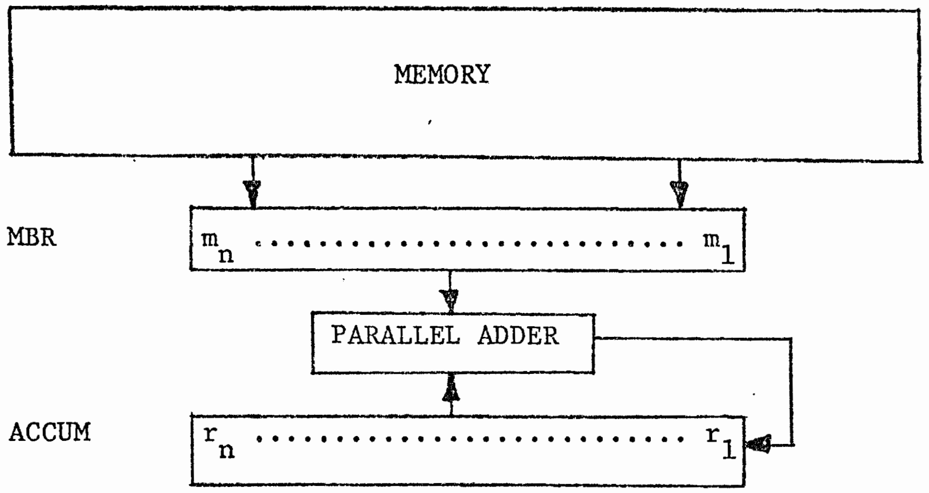
MEMORY
MBR
m_n .... m_1
PARALLEL ADDER
ACCUM
r_n .... r_1

Figure 4. Basic system

Solution

The system suggested for the correction of errors due to single component failures in the adder and in the accumulator is shown in Figure 5. An integer N is represented in a bi-residue code as a triple $[N, |N|_A, |N|_B]$ where $|N|_A$ and $|N|_B$ are the residues modulo A and B respectively and A and B are relatively prime integers. We use the property that the residue of the sum and product of two words is equal to the corresponding sum and product of the residues of two words in the given residue system. When this equality is not maintained, the difference (called the syndrome of the separate code as given in Appendix C) is used to point to a single component failure in either the arithmetic circuitry or the residue checking circuitry (called the checkers, one per residue). Single component failures in the arithmetic circuits are uniquely characterized by discrepancies in both residues of the code; since the residues are formed independently, discrepancy in only one residue indicates a failure in the associated checker unit.

The two residues of a binary word $[m_n, \ldots, m_1]$ are formed (Figure 5)

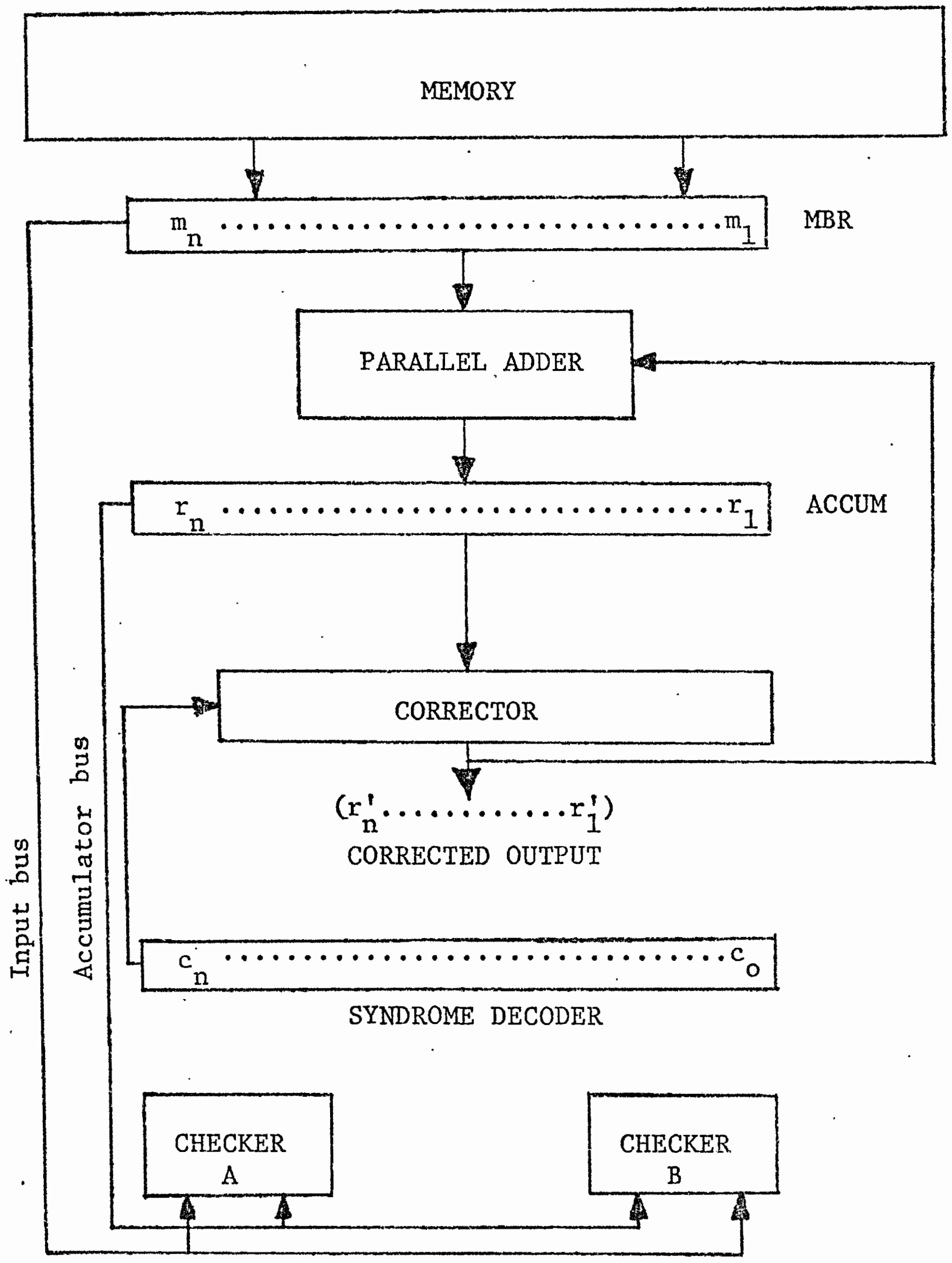

Figure 5. System for correction of single errors

by the checkers A and B; since the words to be added must pass through the MBR, the needed residues are guaranteed. When the ADD instruction is executed, the corresponding residues are added in the checkers, their sum being stored in place of the residue that corresponded to the word that was stored in the ACCUM. After execution of the ADD instruction, the accumulator (ACCUM) contains the sum; the residues of the sum in ACCUM are now formed by the checkers A and B and compared with the sum of the residues. The result of this comparison is the syndrome, which when decoded, gives the sign and magnitude of the error to the corrector. The convention, has been adopted, that if N is the value of the sum if no error occurred; then

$$[r_n, r_{n-1}, \ldots, r_1] = N + e \qquad (4.2)$$

where e defines the sign and magnitude of a possible error, and the contents of the accumulator R represents a binary number. The output of the syndrome decoder, which is also applied to the corrector, $(c_n, \ldots, c_1, c_o)$ is defined as follows:

if $\lvert e\rvert = 0$	$c_i = 0$	for $i=0,1,\ldots,n$
if $\lvert e\rvert = 2^{i-1}$	$c_i = 1 \; c_j = 0$	for $j=1,\ldots,i-1,i+1,\ldots n$
if $e > 0$	$c_o = 0$	
if $e < 0$	$c_o = 1$	

The corrected sum is obtained at the output of the corrector and is denoted $[r'_n, r'_{n-1}, \ldots, r'_1]$.

CHAPTER V

SUMMARY AND CONCLUSIONS

The research reported here considered the mathematical application of codes to the correction of errors originated by malfunctions of the hardware associated with the arithmetic operations of a digital computer. The same principals and applications can be applied to the entire digital communication field, even today with the advanced state-of-the-art, ninety-five percent of the digital data transmissions are intra-computer.

The application of basic error codes to computer arithmetic is analyzed, in some detail, by considering the concept of modular binary arithmetic weight (MBAW). Chapter I relates the concepts of (MBAW) to the arithmetic operations of a finite ring of integers, while defining these operations.

A classification of the arithmetic codes due to Garner [4], points the way to the evaluation of the practical implementation of these codes. In particular, it is found that the AN codes are not very practical, that the non-separate systematic codes are partially modular, and that, the modularity of the separate codes is very desirable. The distance properties of AN codes are discussed at some length which leads to Chapter II.

In Chapter II, the analogy between arithmetic and cyclic communication codes is put on a firm basis; the cyclic large distance codes of Mandelbaum [15] and Barrows [16] are generalized, and a number of examples are given.

The problem of predicting the error correcting properties of separate codes is solved here, when the correspondence between the separate and non-separate codes is established. This correspondence is analyzed in detail, particularly with regard to the possibility of errors also occurring in the modules (checkers) which check the operation of the arithmetic unit. Results are predicted on how many of these checkers may be corrected if they failed at the same time the arithmetic unit fails.

The significance of the above results is that now the theory of the less practical AN codes, which has developed methods of determining their distance properties, may be applied to the multi-residue (separate) codes. The latter are much more attractive for practical implementation because of their much larger range.

The bi-residue single error-correcting codes are the least complex of the multi-residue codes, and the alternatives to their implementation have been considered here. One alternative has been used to show, by Figure 5, a scheme where single errors are corrected with satisfactory results.

This research has dealt only with addition, since it is the basic arithmetic operation used in digital manipulations. This research also has only dealt with the case of the single error. The single error is definitely the most basic of all the possible conditions, but once the error-correction process is proven, it takes no imagination to move directly to the more complex conditions. These are left, along with the remaining arithmetic operations, to further research and to the interested observer now involved with this research.

A SELECTED BIBLIOGRAPHY

1. Massey, J. L., "Survey of Residue Coding for Arithmetic Errors", International Computation Center Bulletin, Vol. 3, Rome, Italy: UNESCO, October, 1964.

2. Peterson, W. W., Error-Correcting Codes, New York: Wiley, 1961.

3. Reitwiesner, G. H., "Binary arithmetic", Advances in Computers, Vol. 1, (F. L. Alt, ed), New York: Academic Press, 1960, pp. 232-308.

4. Garner, H. L., "Error Codes for Arithmetic Operations", IEEE Transactions on Electronic Computers, Vol. EC-15, No. 5, October, 1966. pp. 763-770.

5. Hamming, R. W., "Error Detecting and Error Correcting Codes:, Bell System Technical Journal, Vol. 29, No. 2, April, 1960. pp. 147-160.

6. Rao, T. R. N., "Bi-Residue Error Correcting Codes for Computer Arithmetic", First Southeastern Symposium on System Theory, Conference Record Paper B-3, Virginia Polytechnic Institute, Blacksburg: May 5-6, 1969.

7. Garner, H. L., "A Ring Model for Study of Multiplication for Complement Codes", IEEE Transactions on Electronic Computers, Vol. EC-8, March, 1959. pp. 25-30.

8. Diamond, J. M., "Checking Codes for Digital Computers", Proceedings of the IRE, Vol. 43, April, 1955. pp. 487-488.

9. Brown, D. T., "Error Detecting and Error Correcting Binary Codes for Arithmetic Operations", IRE Transactions on Electronic Computers, Vol. EC-9, September, 1960. pp. 333-337.

10. Szabo, N. S. and Tanaka, R. I., Residue Arithmetic and It's Applications to Computer Technology, New York: McGraw-Hill, 1967.

11. Henderson, D. S., "Residue Class Error Checking Codes", Proceedings of the Sixteenth National Meeting of the Association for Computer Machinery, Los Angeles: 1961.

12. Peterson, W. W., "On Checking an Adder", IBM Journal of Research and Development, Vol. 2, April, 1958. pp. 166-168.

13. Neuman, J. von, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components", Automata Studies, Anals of Mathematics Studies, No. 34, (C.E. Shannon and J. McCarthy, eds.), Princeton University Press, Princeton: 1956. pp. 43-98.

14. Rao, T. R. N., "Error Checking Logic for Arithmetic Type Operations of a Processor", Proceedings of the First Princeton Conference on Information Sciences and Systems, Princeton: 1967.

15. Mandelbaum, D., "Arithmetic Codes with Large Distance", IEEE Transactions of Information Theory, Vol. 1T-13, April, 1967. pp. 237-242.

16. Barrows Jr., J. T., "A New Method for Constructing Multiple Error Correcting Linear Residue Codes", Report R-277, Coordinated Science Laboratory, University of Illinois, Urbana: January, 1966.

17. Hong, S. J., "On Minimum Distance of Multiple Error Correcting Arithmetic Codes", Report R-336, Coordinated Science Laboratory, University of Illinois, Urbana: January, 1967.

18. Chang, S. H. and Tsao-Wu, N., "Distance and Structure of Cyclic Arithmetic Codes (Summary)", Proceedings Hawaii International Conference on System Sciences, (B.K. Kinariwala and F. F. Kuo, eds.), University of Hawaii, Honolulu: 1968. pp. 463-466.

19. Rao, T. R. N., and Garcia, O. N., "Redundancy Techniques for Fault-Tolerant Computers", Interim Report to National Science Foundation, University of Maryland, College Park: 1968.

20. Chien, R. T., "On Linear Residue Codes for Burst Error Correction", IEEE Transactions on Information Theory, Vol. IT-10, April, 1964. pp. 127-133.

21. Chien, R. T., "Burst-Correcting Codes with High Speed Decoding", IEEE Transactions on Information Theory, Vol. IT-15, No. 1, January, 1969.

22. Chien, R. T., "Memory Error Control: Beyond Parity", IEEE, Spectrum; Vol. 10, No. 7, July, 1973.

23. Elias, P., "Computation in the Presence of Noise", IBM Journal of Research and Development, Vol. 2, October, 1958. pp. 346-353.

24. McCoy, N. H., Rings and Ideals, Baltimore: Waverly Press, 1948.

25. McCoy, N. H., The Theory of Rings, New York: McMillian Co., 1965.

26. Northcott, D. G., Ideal Theory, London, England: Cambridge University Press, 1960.

27. Oldham, I. B., Chien, R. T., and Tang, D. T., "Error Detecting and Correction in a Photo-Digital Storage System", IBM Journal of Research and Development, Vol. 12, No. 6, November, 1968.

28. Peterson, W. W., and Weldon, E. J., Error Correcting Codes, Cambridge, Mass: M.I.T. Press, 1972.

APPENDIX A

A DIRECT ALGORITHM TO DETERMINE THE BINARY ARITHMETIC WEIGHT

Algorithms are available for the determination of the arithmetic weight of a number [25],[14]. These algorithms require successive transformation of the number whose arithmetic weight we want to determine and in that sense could, perhaps be called "iterative". An algorithm is presented here that allows the determination of the arithmetic weight of a number without changes in the representation of the number. Its simplicity and the absence of required manipulations make this algorithm superior to the others for either hand or machine computation.

BACKGROUND

The binary arithmetic weight of an integer (BAW), as given in Definition 1.1, is the same as the BAW of its largest odd divisor. (If the integer N is odd, then its largest odd divisor is N itself.) This statement is a direct consequence of the definition of BAW since the low order position with a zero coefficient in the binary expansion of a number does not contribute to its BAW.

Reitwiesner [25] has shown that if N is represented in a form

$$N = \sum_{i=0}^{n} a_i 2^i \tag{1}$$

where a_i is 1, 0, or -1, such that $a_i \cdot a_{i+1} = 0$ for $i = 0, 1, \ldots, n-1$, then the number of non-zero a_i gives the BAW of N. Furthermore, he proved that for any

$$N = \sum_{i=o}^{m} b_i 2^i \qquad (2)$$

where b_i is either 0 or 1, for $i = 0, 1, \ldots, m-1$, and $b_i = i$, a representation of the same integer in form (1), is always possible with $n - m$. Massey [17], more precisely, shows that n, as in (1), will not have to exceed $m + 1$, as in (2), for such a representation.

We will call a run of $(k - j + 1)$ ones a sequence of $b_k = b_{k-1} = \ldots = b_{j+1} = b_j = 1$ for $k > j > = 0$ in the binary representation (2) of N, where no coefficients are missing in the consecutive order from j to k.

Reitwiesner's algorithm to find the BAW of a number N consists in systematically substituting all runs of $(k - j + 1)$ ones that may exist (or originate in the process) starting with the smallest j, by

$$2^{k+1} - 2^j$$

Evidently, this has to be a sequential process since this substitution may create new runs of ones in the higher order positions (if, for example $b_{k+2} = 1$).

This algorithm creates a representation of N of the form (1), as desired, and a count of the non-zero coefficients yields BAW(N).

APPENDIX B

ERRORS IN FINITE RING ARITHMETIC

The binary arithmetic weight of an integer in the infinite ring Z is not affected by the sign of the number, i.e., BAW(N) = BAW(-N) as follows from Definition 1.1.

In finite ring arithmetic modulo m_o the complement of a number N for $0 < N < m$, denoted $\overline{N}$ is $(m_o - N)$. It is clear that $N = -\overline{N}$ modulo m_o. The question arises then, of what is the BAW or a given number N in ring arithmetic; is it BAW(N) of the BAW of its complement with opposite sign? Since both N and its complement with opposite sign are equivalent in ring arithmetic, and we are interested in the minimal binary representation, the following definition is appropriate.

Definition B.1 The modular binary arithmetic weight of an integer N for $0 \leqslant N < m$ in the ring of integers modulo m_o, denoted MBAW(N), is given by

$$\text{MBAW}(N) = \min\ (\ \text{BAW}(N),\ \text{BAW}(\overline{N})\ )$$

Example: In the ring of integers modulo 33 the number N = 17 has MBAW(17) = 1 although BAW(17) = 2, because $\overline{N} = 33 - 17 = 16$ and BAW(16) = 1.

Similarly in finite ring arithmetic the modular binary arithmetic distance between two numbers N_1 and N_2 in the ring is given by $\text{MBAW}(N_1 - N_2)$. This should, of course, be the same as $\text{MBAW}(N_2 - N_1)$. One important property of the binary arithmetic weight of the sum of two integers N_1 and N_2 in the (infinite) ring of integers Z is that

$$\text{BAW}(N_1 + N_2) \leqslant \text{BAW}(N_1) + \text{BAW}(N_2) \qquad \text{(B1)}$$

This is clear if we consider addition of the numbers N_1 and N_2 expressed as $\sum_{i=1}^{n} a_i 2^i$ (where a_i is one of 0, 1, or -1), in a minimum number of terms. Cancellations of non-zero terms or carries may occur, but the number of non-zero terms of the sum cannot possibly exceed the sum of the number of non-zero terms of N_1 and N_2. In a ring of integers modulo m_o it is not true, in general, that

$$\mathrm{MBAW}(|N_1 + N_2|_{m_o}) \leqslant \mathrm{MBAW}(N_1) + \mathrm{MBAW}(N_2) \tag{B2}$$

Example: $m_o = 51$, $N_1 = N_2 = 32$

$$\mathrm{MBAW}(|64|_{51}) = \mathrm{MBAW}(13)$$
$$= \min(\mathrm{BAW}(13), \mathrm{BAW}(38)) = 3$$

while MBAW(32) = 1 and the contradiction of (B2) follows.

<u>LEMMA B1.1</u> In conventional binary number systems when $M_o = 2$ (i.e., in a radix binary number system) or when $m_o = 2^n - 1$ (i.e., in a diminished radix binary number system), the inequality (B2) is valid.

<u>PROOF</u>: Let N_i' be either N_i or $-\overline{N}_i$ for i = 1, 2, correspondingly, whichever has the smallest BAW. From elementary modular addition we can say:

$$|N_1 + N_2|_{m_o} = |N_1' + N_2'|_{m_o}$$

Then $$\mathrm{MBAW}(|N_1 + N_2|_{m_o}) = \mathrm{MBAW}(|N_1' + N_2'|_{m_o})$$

and $$\mathrm{MBAW}(|N_1' + N_2'|_{m_o}) < \mathrm{MBAW}(|N_1' + N_2'|_{m_o})$$

as a consequence of Definition B.1.

In a sum, a carry into the (n + 1) bit position when $m_o = 2^n$ will be neglected since it has magnitude 2^n and $|2^n|_{m_o} = 0$ in this case. If

$m_o = 2^n - 1$ a carry into the (n + 1) bit position will be equivalent to adding 1 to the sum (often called an "end around carry") while removing the bit from the (n + 1) position, since now $|2^n|_{m_o} = 1$ for $m_o = 2^n - 1$.

Consequently, in a radix binary system the possible extra carry is lost and in a diminished radix binary system it is carried around to the least significant bit position. In either case $N_1' + N_2'$ does not increase its BAW by taking $|N_1' + N_2'|_{m_o}$. Then:

$$BAW(|N_1' + N_2'|_{m_o}) \leqslant BAW(N_1' + N_2')$$

but by (B1) above

$$BAW(N_1' + N_2') \leqslant BAW(N_1') + BAW(N_2')$$

and because of our choice of N' for i = 1, we have

$$BAW(N_1') = MBAW(N_i)$$

for i = 1, 2 which proves (B2).

When an error pattern E occurs in the transmission of an element N_1 of a ring of integers modulo m_o or in the addition of two elements of the ring which would have had a correct result N_1, then the erroneous result N_2 is given by

$$N_2 = |N_1 + E|_{m_o} \qquad (B3)$$

where E is also restricted to $0 \leqslant E < m$ with the understanding that E = 0 only if there is no error.

<u>Definition B.2</u> The set of all error patterns E in the ring of integers modulo m_o co-responding to all errors in the ring, of MBAW less than or equal to d is denoted by $U(m_o,d)$.

More formally

$$U(m_o,d) = \{E \mid 0 \leqslant E < m_o \text{ and } MBAW(E) = d\} \qquad (B4)$$

LEMMA B1.2 If by $U(m_o,d_1) \dot{+} U(m_o,d_2)$ we mean the set formed by the sum modulo $m_o = 2^n - 1$, for some n, of each of the elements of $U(m_o,d_1)$ with each of the elements of $U(m_o,d_2)$ in all possible ways, then we can establish the equality of the sets:

$$U(m_o,d_1) \dot{+} U(m_o,d_2) = U(m_o,d)$$

where $d = \min\ (d_1 + d_2,\ [\frac{n}{2}])$ and $[\frac{n}{2}]$ is the integer part of $\frac{n}{2}$.

PROOF: For any $E_1 \varepsilon U(m_o,d_1)$ and $E_2 \varepsilon U(m_o,d_2)$ by Lemma B1.1

$$\text{MBAW}(|E_1 + E_2|_{m_o}) = \text{MBAW}(E_1) + \text{MBAW}(E_2) = d_1 + d_2$$

The maximum possible MBAW of any n bit word is the integer part of $\frac{n}{2}$, denoted $[\frac{n}{2}]$. If $d_1 + d_2 < [\frac{n}{2}]$ then $|E_1 + E_2|_{m_o} \varepsilon U(m_o,d_1 + d_2)$; otherwise if $d_1 + d_2 \geqslant \frac{n}{2}$, then $|E_1 + E_2|_{m_o} \varepsilon U(m_o,[\frac{n}{2}])$ which contains all possible error patterns.

Conversely any element of $U(m_o,d_1 + d_2)$ for $d_1 + d_2 \leqslant [\frac{n}{2}]$ may be decomposed in the sum modulo m_o of an element from $U(m_o,d_1)$ and another from $U(m_o,d_2)$.

APPENDIX C

NECESSARY AND SUFFICIENT CONDITIONS FOR ERROR CORRECTION

Peterson [2] has given the following:

<u>THEOREM C.1</u> An AN binary code is capable of single error correction for all numbers in the range $0 \leqslant N < M$ if and only if the residues of $\pm 2^i$ modulo A are distinct and non-zero for all i such that 2^i - AM.

This result can then be extended.

<u>Definition C1.1</u> The syndrome of a non-separate code word AN which has been corrupted by an error pattern E is defined by

$$S(AN_1 + E) = |AN_1 + E|_A = |E|_A \tag{C1.1}$$

The syndrome of a separately coded word $X = [x, x_1, \ldots, x_k]$, where x is the ring of integers modulo m_o, is defined by the k-tuple

$$S(X) = (s_1, \ldots, s_k) \tag{C1.2}$$

where $s_i = |x - x_i|_{m_i}$ for $i = 1, 2, \ldots, k$.

Consider Z, the correct result of the addition of multi-residue coded words. If an error of pattern E has occurred in the arithmetic corresponding to the first component of the word Z, then clearly

$$S(Z + E) = (|E|_{m_i}, \ldots, |E|_{m_k}) \tag{C1.3}$$

In either kind of code, separate or non-separate, the necessary and sufficient condition for the detection of all error patterns E of positive weight no greater than d is that

$$S(X + E) \neq S(0) \tag{C1.4}$$

for all $E \neq 0$ in $U(m_o,d)$, and any code word X.

Similarly, the necessary and sufficient conditions for the correction of all error patterns E of positive weight on greater than d is that $S(X + E_1) \neq S(X + E_2)$ for any pair $E_1 \neq 0$ and $E_2 \neq E_1$ in $U(m_o,d)$, and any code word X.