Reprinted from

ADVANCES IN INFORMATION SYSTEMS SCIENCE

Vol. 4

(Ed. J. T. Tou)

New York and London: Plenum, 1972.

ERROR-CORRECTING CODES IN COMPUTER ARITHMETIC

James L. Massey
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana

and

Oscar N. García
College of Engineering
University of South Florida
Tampa, Florida

1. INTRODUCTION

This chapter is intended to summarize the most important results which have been obtained in the theory of coding for the correction and detection of errors in computer arithmetic. The rapid growth in the size and speed of digital computers has placed stringent reliability demands on the arithmetic unit. Attempts to satisfy these demands have generally followed one of three directions: (1) Attempts to improve the reliability of the components used in the construction of the arithmetic unit, (2) attempts to improve reliability by incorporating hardware redundancy so that the result of a computation is unaffected by the failure of one or more of the replicated units which form the arithmetic unit, or so that the failure of one or more of the replicated units can be detected and the faulty units replaced, and (3) attempts to incorporate redundancy into the numbers themselves which are being processed so that erroneous results can be corrected or detected. This third approach, which is the subject of this chapter, tacitly assumes that it is possible to build the "decoder" which corrects or detects erroneous results much more reliably than the arithmetic unit which it monitors, so that the decoder can be considered error-free for practical purposes.

Because of its dominating practical importance, only binary arithmetic will be considered in this chapter. This restriction allows a simplification

in the notation and proofs to be given, but we should remark that many results can be easily extended to an arbitrary radix.

The two most important types of coding which have been proposed for use in arithmetic units are the AN-codes and the multiresidue codes. The precise mathematic definition of these codes will be given later, but a general description at this point may help the reader to place the subsequent detailed analysis into perspective. In an AN-code, the integer N is "encoded" by multiplication with a fixed integer A before being represented in the computer. Thus, only a fraction $1/A$ of the possible representations are ever intentionally used in the computer; the resultant redundancy can then be used for error correction and detection. In a multiresidue code, the integer N is first encoded into $k+1$ distinct integers $[N, N_1, \ldots, N_k]$, where each N_i is the residue of N modulo some fixed integer m_i. Each of these $k+1$ integers is then represented separately in the computer. The redundancy in this type of coding is obvious. Whereas AN-codes and multiresidue codes may seem superficially quite disparate, there is in fact an intimate connection between these two forms of coding, as will be shown in Section 4.

Our specific goal in this chapter is to provide a self-contained introduction to AN-codes and multiresidue codes that is at the same time comprehensive of the most important known results in this area. Complete proofs of most assertions will be given, both for the insight which these proofs provide in many cases and also because the proofs are often not to be found in readily available literature or as usually given are much more complicated than the proofs which we give here.

1.1. Arithmetic Weight

An important first step in the design of any coded system is the identification of the likely modes of failure in the system and the formulation of a convenient description of the resultant errors in its output. We shall now examine these questions in detail for a binary adder, i.e., for a unit which performs addition on the radix-2 forms of integers. For conceptual simplicity at this point, we impose no magnitude restriction on the integers to be added except that they be nonnegative. Note that this causes the adder to be a potentially infinite device since the integers which it must add could have arbitrarily many nonzero terms in their radix-2 forms.

The radix-2 form of an integer I is the expression for I of the type

$$I = a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \cdots + a_i 2^i + \cdots \tag{1}$$

where each coefficient a_i is either 0 or 1. When $a_i = 0$ for $i \geq n$, we shall

often write the radix-2 form as the n-tuple $[a_{n-1}, \ldots, a_1, a_0]$. If also $a_{n-1} \neq 0$, we shall say that n places are required to represent the radix-2 form for I. For example, $I = 11$ has the 5-tuple radix-2 form [01011], but only four places are required to represent the radix-2 form for 11.

The (potentially infinite) circuitry for adding two integers in radix-2 form can be thought of as a sequence of elemental adding units, associated with each power i of 2, each of which forms the modulo-2 sum of its inputs, which are the coefficients of 2^i in each of the radix-2 forms being added and the carry bit from the previous unit, and generates a carry bit to the next unit. This flow of carries is shown by the graph in Fig. 1, whose nodes correspond to the elemental adding unit which adds the coefficients of the designated power of two.

For the situation shown in Fig. 1, it is quite natural to think of a "single error" in the adder as constituting an incorrect modulo-two sum formed by one adding unit or the generation of an incorrect carry bit by one adding unit. If the failure is in the unit denoted 2^i in Fig. 1, then an error of the former type would alter the correct sum by $+2^i$ or -2^i, whereas an error of the latter type would alter the correct sum by $+2^{i+1}$ or -2^{i+1}. Thus, a single error always causes the erroneous sum to differ from the correct sum by $+2^j$ or -2^j for some j. If the erroneous sum differs from the correct sum by an amount E, it is then natural to think of the weight of E, i.e. the number of errors, as the least number of terms of the form $+2^j$ or -2^j whose sum is E. E could not have resulted from fewer than this many single errors occurring together and there is some combination of this many single errors occurring together which give an error E. This measure of the weight of an error is called "arithmetic weight" and will now be formalized.

An expression for an integer E (which may be positive, negative, or zero) of the type

$$E = e_0 + e_1 2^1 + e_2 2^2 + \cdots + e_i 2^i + \cdots \tag{2}$$

where each coefficient e_i is -1, 0, or $+1$, is called a *modified binary form* for E. In general, the modified binary form of an integer is not unique, e.g., $E = 1$ has the two modified binary forms 2^0 and $-2^0 + 2^1$ and in fact infinitely many other modified binary forms. We do note, however, from (2) that if E is divisible by 2^i, then $e_0 = e_1 = \cdots = e_{i-1} = 0$ in every

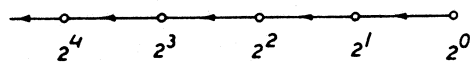

Fig. 1. Graph showing carry flows in the integer addition of radix-2 forms.

modified binary form of E. It follows that $E = 0$ has the unique modified binary form with all zero coefficients.

Definition 1.1. The *arithmetic weight* of the integer E, denoted $W(E)$, is the least number of nonzero coefficients in a modified binary form for E.

Although the modified binary form of an integer is not unique in general, there is a particular modified binary form introduced by Reitwiesner (1) which is unique and which has proved of much importance in arithmetic coding, called the *nonadjacent form*. A modified binary form is said to be a nonadjacent form (or NAF for short) if the coefficients e_i in (2) satisfy $e_i e_{i+1} = 0$ for $i = 0, 1, 2, \ldots$, in other words, if there are no two adjacent nonzero coefficients in the form.

Lemma 1.1. Every integer E has at most one nonadjacent form.

Proof. Let e_i and e_i' denote the respective coefficients in two nonadjacent forms for E. If the NAF's with coefficients e_i and $-e_i'$ are added term by term with carries to the following or preceding term when $e_i = -e_i' = 1$ or $e_i = -e_i' = -1$, respectively, then the carries cannot propagate, since the nonadjacency of each form requires that both have zero terms following and preceding the term which generated the carry. Hence, the addition so performed yields directly a modified binary form for $E + (-E) = 0$ which is the unique form with all zero coefficients. But this summation will give all zero coefficients only if $e_i = e_i'$ for all i.

The following theorem, due to Chang and Tsao-Wu (2) shows that every integer does indeed have an NAF and also provides what is generally the simplest method for the hand calculation of the NAF.

Theorem 1.1. Let $[b_{n-1}, \ldots, b_1, b_0]$ and $[a_{n-1}, \ldots, a_1, a_0]$ be the radix-2 forms of the positive integers $3E$ and E respectively; then, the coefficients e_i defined by

$$\begin{aligned} e_i &= b_{i+1} - a_{i+1} && \text{for } i = 0, 1, \ldots, n-2 \\ e_i &= 0 && \text{for } i \geq n-1 \end{aligned}$$

are the coefficients in the NAF for E.

Proof. We note first that $3E$ and E are both even or both odd, so that $b_0 = a_0$. Hence,

$$\begin{aligned} \sum_{i=0}^{n-2} e_i 2^i &= \sum_{i=0}^{n-2} (b_{i+1} - a_{i+1}) 2^i \\ &= \tfrac{1}{2} \sum_{i=0}^{n-1} (b_i - a_i) 2^i \\ &= \tfrac{1}{2}(3E - E) = E \end{aligned}$$

so that the e_i are certainly the coefficients in a modified binary form for E and it remains to show that this form is nonadjacent, i.e., that $e_i e_{i+1} = 0$ for all i. If $e_i = 0$, there is nothing to show. Suppose then that $e_i = -1$, i.e., that $b_{i+1} = 0$ and $a_{i+1} = 1$. We next note that we can obtain $3E$ by adding the radix-2 forms for E and $2E$, the latter being just $[a_{n-2}, \ldots, a_0, 0]$. Hence, the coefficients in the radix-2 form of $3E$ are given by

$$b_{i+1} = a_{i+1} + a_i + c_i \pmod 2$$

where c_i is the carry bit from the addition which gave b_i. But by hypothesis, $b_{i+1} = 0$ and $a_{i+1} = 1$, so that we must have

$$a_i + c_i = 1 \pmod 2.$$

Thus, two of the bits being added to give b_{i+1} must be one's so that the next carry $c_{i+1} = 1$. But this in turn implies

$$\begin{aligned} b_{i+2} &= a_{i+2} + a_{i+1} + c_{i+1} \pmod 2 \\ &= a_{i+2} \end{aligned}$$

so that $e_{i+1} = b_{i+2} - a_{i+2} = 0$ and hence $e_i e_{i+1} = 0$, as was to be shown. The argument when $e_i = +1$ is entirely similar.

In words, Theorem 1.1 states that the NAF of E can be obtained by the bit-by-bit subtraction of the radix-2 forms for $3E$ and E, when E is a positive integer. If E is a negative integer, its NAF can be obtained by changing the signs of the coefficients in the NAF of $-E$.

Example 1.1. $E = 187$. $3E = 561$.

$$\begin{aligned} \text{Radix-2 form for } 3E &= [1\,0\,0\,0\,1\,1\,0\,0\,0\,1] \\ \text{Radix-2 form for } E &= [0\,0\,1\,0\,1\,1\,1\,0\,1\,1] \\ \hline \text{NAF for } E &= [1\,0\,\bar{1}\,0\,0\,0\,\bar{1}\,0\,\bar{1}] \end{aligned}$$

where here and hereafter we write $\bar{1}$ for -1 when writing an NAF as an n-tuple, i.e., as $[e_{n-1}, e_{n-2}, \ldots, e_1, e_0]$. If $e_{n-1} \neq 0$, we shall say that n places are required for the NAF of E.

As an immediate consequence of Theorem 1.1, we have the following:

Corollary 1.1. The number of places required for the NAF of E (E a positive integer) is one less than the number of places required for the radix-2 form of $3E$. In particular, the number of places required for the NAF of E is at most one greater than the number of places required for the radix-2 form of E.

For our purposes, the most important property of NAF's is that given in the following theorem due to Reitwiesner ([1]).

Theorem 1.2. For any integer E, there are $W(E)$ nonzero coefficients in the NAF for E, i.e., no modified binary form for E has fewer nonzero coefficients than the NAF for E.

This theorem can be proved with a slight modification of the argument used to prove Lemma 1.1, namely by supposing that the e_i are the coefficients in the NAF for E and that the e_i' are the coefficients in a modified binary form for E (not an NAF) with fewer nonzero coefficients, then showing that the digit-by-digit addition of the e_i and the $-e_i'$ with carries (which may now propagate more than one position) must give a modified binary form with a least one nonzero coefficient so that the sum does not equal zero, thus contradicting the assumption that the e_i and the e_i' represent the same number. We leave it to the interested reader to supply the details of the proof.

For the convenience of the reader who might wish to program the calculation of NAF's for a computer, we restate the method for their calculation given by Theorem 1.1 as the following algorithm.

Algorithm 1.1

Step (0). Let $[a_{n-1}, \ldots a_1, a_0]$, be the radix-2 form for a positive integer E such that $a_{n-1} = a_{n-2} = 0$. Set $i = 0$ and $c_0 = 0$.

Step (1). Calculate in order

$$b_{i+1} = a_{i+1} + a_i + c_i \pmod 2$$

$$c_{i+1} = \text{carry bit from the above addition}$$

$$e_i = b_{i+1} - a_{i+1}$$

Step (2). If $i = n - 2$, stop. Otherwise, increase i by 1 and return to step (1). When the calculation has stopped, $[e_{n-2}, \ldots, e_1, e_0]$ is the desired NAF for E.

It is interesting to note that Reitwiesner's original study of NAF's was not motivated by error correction but by the desire to minimize the number of additions and subtractions performed in compound arithmetic operations such as multiplication and division. Reitwiesner himself gave an algorithm for calculating the NAF for E starting from any modified binary form for E, and modifications of this algorithm have been given by Bernstein ([3]) and Barrows ([4]).

When one is interested only in $W(E)$, it is not necessary to make the entire calculation to find the NAF of E. The following algorithm, due to García ([5]) and directly inferable from algorithm 1.1, permits the direct calculation of $W(E)$ from the radix-2 form for E.

Algorithm 1.2

Step (0). Let $[a_{n-1}, \ldots, a_1, a_0]$ be the radix-2 form for a positive integer E such that $a_{n-1} = a_{n-2} = 0$. Set $i = 0$, $W = 0$, and $a_{-1} = 0$.

Step (1). If $a_i = a_{i-1}$, go to step (2). Otherwise, increase W by 1, and increase i by 1.

Step (2). If $i \geq n - 2$, stop. Otherwise, increase i by 1 and go to step (1). When the calculation has stopped, $W = W(E)$.

It should also be mentioned that Chiang and Reed ([6]) have given an algorithm for calculating $W(E)$ that does not require a radix-2 form for E as a starting point but works directly with integers. Also, Goto and Fukumura ([7]), and Chien *et al.* ([8]) have given algorithms for the computation of the NAF of E based upon nonrestoring binary division.

1.2. Arithmetic Distance

For any two integers E and E', the following properties of a *norm* are satisfied by arithmetic weight:

$$W(E) = W(-E) \tag{3}$$

$$W(E) \geq 0 \quad \text{with equality if and only if} \quad E = 0 \tag{4}$$

$$W(E + E') \leq W(E) + W(E') \tag{5}$$

Properties (3) and (4) follow directly from Definition 1.1. Property (5) is proved by noting that if the NAF's of E and E' are added digit-by-digit, then carries can flow only into the adjacent terms where the digit-by-digit sum is zero, hence $E + E'$ can certainly be written as a modified binary form (which may not be an NAF) with at most as many nonzero terms as the sum of the number of nonzero terms in the two NAF's being added. But $W(E + E')$ is the least number of terms in any modified binary form for $E + E'$, so that (5) follows.

The norm properties (3)–(5) of arithmetic weight guarantee that a true metric for the set of integers can be defined as follows:

Definition 1.2. The *arithmetic distance* between the integers I_1 and I_2, denoted $D(I_1, I_2)$, is the arithmetic weight of their difference, i.e.,

$$D(I_1, I_2) = W(I_1 - I_2) \tag{6}$$

It is readily checked that for any integers I_1, I_2, and I_3, the following three properties which formally define a *metric* are satisfied by arithmetic distance:

$$D(I_1, I_2) = D(I_2, I_1) \tag{7}$$

$$D(I_1, I_2) \geq 0 \quad \text{with equality if and only if} \quad I_1 = I_2 \tag{8}$$

$$D(I_1, I_2) \leq D(I_1, I_3) + D(I_3, I_2) \tag{9}$$

These properties are called the symmetry, positive-definite, and triangle-inequality properties, respectively. Their proofs may be obtained directly from (3), (4), and (5) respectively.

It follows from (7)–(9) that any set of integers with arithmetic distance taken as the measure of "distance" forms a true metric space. As we shall see in Section 2, this fact plays an important role in the study of codes for the correction of errors in integer addition.

Finally, suppose that the correct result of some operation is the integer J but the actual result is $I = J + E$, where E is an error with $W(E) = i$. It follows from (6) that

$$D(I, J) = i$$

i.e., that an error of weight i in some result causes the erroneous result to be at arithmetic distance i from the true result. We shall make considerable use of this fact in the sequel.

1.3. Errors in Computer Arithmetic

We have seen that arithmetic weight is an appropriate measure of errors in the ordinary addition of integers. Our main interest in this chapter, however, is not in the ordinary arithmetic of integers but in the types of arithmetic commonly used in digital computers, i.e., the arithmetic of integers modulo m, where m is either $2^n - 1$ (so-called one's complement arithmetic) or 2^n (so-called two's complement arithmetic.) These types of arithmetic will be considered in greater detail in Section 2.3, but we wish here to point out that the carry flows in the circuitry for performing addition modulo m on the radix-2 forms of integers are as shown in the graphs in Fig. 2 for $m = 2^n - 1$ and $m = 2^n$.

If $[a_{n-1}, \ldots, a_1, a_0]$ and $[b_{n-1}, \ldots, b_1, b_0]$ are the radix-2 forms of the integers to be added, then the nodes labeled 2^i in Fig. 2 denote the elemental adding units which perform the modulo-two addition of a_i, b_i and the carry bit into that node and which generate a carry bit of 1 out of the node if two or more of these digits are 1's. From Fig. 2, we see immediately that the arithmetic weight of an error has the same significance for these two types of addition as it has for the ordinary addition of integers, namely that t failures in the elemental adding units (each failure being an incorrect modulo-two sum or an incorrect carry bit) still creates an error E of arithmetic weight at most t in the sum. This follows from the fact that the carry bit generated by any elemental adding unit flows to at most one other unit. (It should be pointed out of course that if, in an elemental adding unit, the logic which performs the modulo-two addition is shared with that which generates the carry bit, then a single logical malfunction can create an error of arithmetic weight two. This remark applies to the ordinary addition of integers as well as the modulo m addition for $m = 2^n - 1$ or $m = 2^n$.) Thus, the arithmetic weight of an error is still an appropriate measure of the amount of logical malfunction in the adder just as it was for the ordinary addition of integers in radix-2 form.

To show that arithmetic weight is not in general an appropriate measure of adder malfunction for moduli m other than $2^n - 1$ and 2^n, we point to the graph in Fig. 3, which shows the carry flows in the modulo 11 addition of integers in radix-2 form. Since

$$2(2^3) = 5 = 2^2 + 2^0 \pmod{11}$$

it follows that an error in the carry from the leftmost adding unit in Fig. 3 results in an error of arithmetic weight two in the sum.

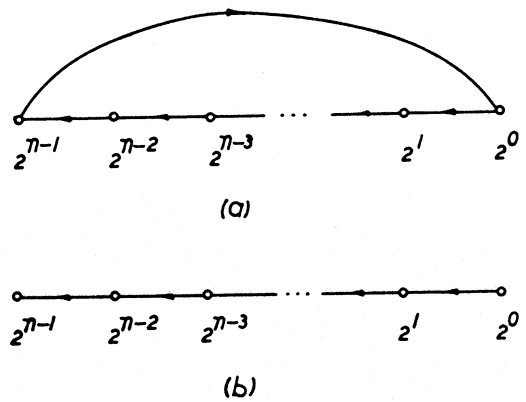

Fig. 2. Graphs showing carry flows in the modulo m addition of radix-2 forms: (a) $m = 2^n - 1$ and (b) $m = 2^n$.

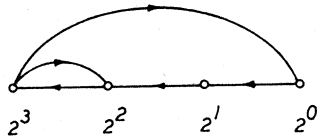

Fig. 3. Graph showing carry flows in the modulo 11 addition of radix-2 forms.

2. THE GENERAL THEORY OF AN-CODES

Definition 2.1. The *AN-code* generated by a positive integer A is the set of integers AN for $0 \leq N < B$, where B is a specified integer which determines the number of integers in the code.

The integers in an AN-code will be called the *code points*. The code point AN is to be thought of as the encoded version of the "information integer" N. Note that $AN_1 + AN_2 = A(N_1 + N_2)$ so that the ordinary sum of the code points for N_1 and N_2 is the code point for the sum $N_1 + N_2$ provided that $N_1 + N_2 < B$, i.e., provided that the sum does not overflow from the code. Hence, an AN-code is "linear" in this restricted sense, and for this reason AN-codes are sometimes called "linear residue codes."

2.1. Minimum Arithmetic Distance

Definition 2.2. The *minimum arithmetic distance* $D_{\min}$ of an AN-code is the minimum of the arithmetic distances between all pairs of distinct code points in the code. By way of convention, $D_{\min} = \infty$ when there is only one code point in the code.

A simple consequence of the "linearity" of an AN-code is given in the following theorem:

Theorem 2.1. The minimum arithmetic distance $D_{\min}$ of an AN-code coincides with the minimum arithmetic weight $W_{\min}$ of the $B - 1$ nonzero code points in the code, i.e.,

$$D_{\min} = W_{\min} = \min_{0<N<B} W(AN) \tag{10}$$

Proof. If AN_1 and AN_2 are code points with $N_1 > N_2$, then

$$D(AN_1, AN_2) = W[A(N_1 - N_2)] = W(AN_3)$$

where $0 < N_3 = N_1 - N_2 < B$, which shows that the distance between any pair of distinct code points is always the weight of some nonzero code point. Conversely, if AN_3 is any nonzero code point, then

$$D(0, AN_3) = W(AN_3)$$

so that the weight of every nonzero code point is also the distance between some pair of distinct code points and the theorem is proved.

2.2. Error Correction in Integer Arithmetic

We now relate the minimum arithmetic distance of an AN-code to its ability to correct and/or detect errors in its code points. In this and all subsequent similar discussions, we assume that the correct result of some operation is some code point in the AN-code and that any code point is a possible result. For example, the operation might be the addition (without overflow from the set of code points) of the code points in the code. If the correct result is the code point AN but the actual result is $I = AN + E$, we shall say that there is an error of weight $W(E)$ in the result and we shall call E the *actual error*. The decoder, i.e., the device which does the actual error correction and/or detection, is allowed to operate only on the actual result I. The basic theorem which governs the error correcting and detecting ability of an AN-code when the actual result I may be any integer is the following:

Theorem 2.2. For any $t \geq 0$ and any $s \geq 0$, an AN-code can correct all errors of weight t or less in its code points and can detect all other errors of weight $t + s$ or less in its code points if and only if its minimum arithmetic distance satisfies

$$D_{\min} > 2t + s$$

Proof. Suppose that $D_{\min} > 2t + s$. We shall show that the decoder which decodes the result I into the nearest code point AN_1 if there is a code point AN_1 with $D(I, AN_1) \leq t$, and announces a detected error otherwise, performs the claimed correction and detection. For, suppose $I = AN + E$, where AN is the correct result and $W(E) \leq t$. Then, $D(I, AN) \leq t$ but for any other code point AN', we have by the triangle inequality (9), that $D(I, AN') \geq D(AN, AN') - D(I, AN) > t + s$, so that $N_1 = N$ and the error is corrected as claimed. Similarly, if $t < W(E) \leq t + s$, then for any other code point AN', $D(I, AN') > t$, so that no code point within distance t of I exists and a detected error will be announced as claimed.

Conversely, suppose that $D_{\min} \leq 2t + s$. Then, there exist distinct code points AN and AN' such that $D(AN, AN') \leq 2t + s$. Let $AN - AN'$

$= E' - E$, where E' is the sum of the t highest-order terms in the NAF of $AN - AN'$ and $-E$ is the sum of the remaining terms. Hence, $W(E') \leq t$ and $W(E) \leq t + s$. Now, if the actual result is

$$I = AN + E = AN' + E'$$

a decoder to perform the claimed correction and detection would be inconsistently required to correct E' and also to correct or detect E. Hence, a decoder to perform the claimed correction and detection does not exist, and the theorem is proved.

Since our interest in this chapter is the correction of errors in computer arithmetic, we shall not further consider AN-codes for correcting errors in unrestricted integer arithmetic. We refer the reader with further interest in this latter subject to an earlier survey by Massey ([9]) which treats this case extensively.

2.3. Arithmetic Modulo m

We have previously pointed out that the arithmetic in most digital computers is arithmetic modulo m where either $m = 2^n - 1$ (one's complement arithmetic) or $m = 2^n$ (two's complement arithmetic). In this section, we give the necessary mathematical preliminaries required for the study of error correction in such arithmetic.

For any positive integer m, the *residue* of an integer I modulo m is defined to be the remainder when I is divided by m. The residue of I modulo m will be denoted hereafter as $R_m(I)$. In other words, $R_m(I)$ is the unique integer in the range $0 \leq R_m(I) < m$ such that $I - R_m(I)$ is divisible by m. For example, $R_{11}(15) = R_{11}(-7) = 4$. For our purposes, the two most important properties of residues are

$$R_m(I_1 \pm I_2) = R_m[R_m(I_1) \pm R_m(I_2)] \tag{11}$$

and

$$R_m(I_1 I_2) = R_m[R_m(I_1) R_m(I_2)] \tag{12}$$

which say essentially that if only the residue of the final result is of interest, then the residues of all intermediate results can be taken without changing the final result.

If for the set of integers $\{0, 1, \ldots, m-1\}$ we define an additon operation, which hereafter we denote by the "ring sum" sign $\oplus$, by

$$I_1 \oplus I_2 = R_m(I_1 + I_2) \tag{13}$$

and we define a multiplication operation, which hereafter we denote by the "ring times" sign $\otimes$, by

$$I_1 \otimes I_2 = R_m(I_1 I_2) \tag{14}$$

then $Z_m = \{0, 1, \ldots, m-1\}$ under these operations becomes an algebraic structure known as a *ring* and is called "the ring of integers modulo m." Arithmetic in this ring is called "arithmetic modulo m."

For any integer $I \neq 0$ in Z_m, its "negative" is the integer $m - I$ since $I \oplus (m - \mathrm{I}) = 0$. Hereafter, we use the symbol $\ominus$ ("ring minus") for the minus sign in Z_m and we note that $\ominus I = I$ for $I = 0$ and that $\ominus I = m - I$ for $I \neq 0$. Following the usual convention for plus and minus signs, we shall write $I_1 \ominus I_2$ rather than the more complete $I_1 \oplus (\ominus I_2)$.

For the special case where m is a prime p, the ring Z_p becomes a *field*, and each nonzero I in Z_p has a multiplicative inverse, denoted I^{-1}, in Z_p such that $I \otimes I^{-1} = 1$. In this case, Z_p is called the finite or Galois field of p elements and is usually denoted GF(p) rather than Z_p.

2.4. Errors in Addition Modulo m

In an arithmetic unit which performs arithmetic modulo m, we shall always assume that the actual result of any operation is an integer I such that $0 \leq I < m$, i.e., that I is an integer in Z_m. Moreover in considering AN-codes for error correction and detection in addition modulo m, we shall always assume that the generator A and the number of code points B in the code are related by

$$m = AB \tag{15}$$

Then, if AN_1 and AN_2 are any code points, and hence necessarily integers in Z_m, we have

$$AN_1 \oplus AN_2 = R_{AB}[AN_1 + AN_2] = AR_B(N_1 + N_2)$$

and since $0 \leq R_B(N_1 + N_2) < B$, it follows that the modulo m sum of any two code points is another code point. Thus, an AN-code is truly linear for modulo m addition. (Not only are the code points closed under addition modulo m and thus the code is a group, but the modulo m product of a code point with any integer in Z_m is another code point so that the code is in fact an *ideal* in Z_m.)

If the code point AN is the correct result of some operation in Z_m,

but the actual result is I, we define the *ring error* F as the unique element of Z_m such that $I = AN \oplus F$, or equivalently, $F = I \ominus AN$. Note that the actual error E is defined as $I = AN + E$ so that the actual error E is either F or $F - m$ depending on whether $E \geq 0$ or $E < 0$, respectively. Note also that the actual error may not be an integer in Z_m.

The *syndrome* $S(I)$ associated with the possibly erroneous result $I = AN \oplus F$ is defined to be the residue of I modulo A, the generator of the code, i.e., $S(I) = R_A(I)$. Since $m = AB$, we have

$$S(I) = R_A(I) = R_A(AN \oplus F) = R_A(F) \tag{16}$$

and we see that the syndrome depends only on the ring error which has occurred, and we commonly speak of $R_A(F)$ as the syndrome of the ring error F. The importance of the syndrome is that it uniquely identifies which ring errors are consistent with the result I.

Lemma 2.1. For any I in Z_m, the set of ring errors consistent with I (i.e., the set of integers F in Z_m such that $I = AN \oplus F$ for some code point AN) is the set of ring errors whose syndrome is $S(I)$. This set contains precisely B distinct ring errors.

Proof. The possible ring errors are the distinct F_i such that

$$I = Ai \oplus F_i, \qquad 0 \leq i < B$$

so there are precisely B such ring errors. Moreover $S(I) = R_A(Ai \oplus F_i) = R_A(F_i) = S(F_i)$ for all of these B ring errors. The A different possible values for $S(I)$ together with the B ring errors as given above with the syndrome $S(I)$ account for all $m = AB$ ring errors in Z_m, so no ring errors except the F_i given above can have the same syndrome $S(I)$.

In coding for arithmetic in Z_m, one generally first specifies a set T_c of ring errors which one desires to correct. That is, for an error F in T_c and a result such that $I = AN \oplus F$, one wishes the decoder to correct the result I to its true value AN. In practice, one would always assign $F = 0$ to T_c since one would certainly wish the decoder to pass correct results unchanged to the user, but this assumption is not necessary in the following theory. Similarly, one generally specifies another set T_d of ring errors (which may be the empty set) disjoint from T_c which one desires to detect. That is, for an error F in T_d and a result I such that $I = AN \oplus F$, one wishes the decoder to announce a detected but uncorrectable error. The following theorem gives the necessary and sufficient conditions for determining whether such error correction and detection is possible for a given AN-code.

Theorem 2.3. Let T_c and T_d be any disjoint subsets of Z_m. Then, an AN-code (with $m = AB$) can correct all ring errors in T_c in its code points and can detect all ring errors in T_d in its code points if and only if there is no ring error F_1 in T_c which has the same syndrome as some other ring error F_2 in either T_c or T_d.

Proof. If the distinctness condition on the syndromes holds, and if $I = AN \oplus F$ for some F in T_c, then Lemma 2.1 implies that no other error F' in T_c or T_d is possible so that I can be unambiguously decoded to AN or a detected error unambiguously announced if there is no such F in T_c. Conversely, if the distinctness condition does not hold, i.e., if $S(F_1) = S(F_2)$ for some F_1 in T_c and some other F_2 in T_c or T_d, then by Lemma 2.1, both F_1 and F_2 are possible errors for the result $I = F_1$ and hence a decoder would be inconsistently required to correct F_1 and to correct or detect F_2.

Our proof of Theorem 2.3 suggests a possible error-correcting procedure for AN-codes used in modulo m arithmetic. From the actual result I, one could first compute the syndrome $S(I)$ and then look up the corresponding ring error F (if any) in T_c in a table of syndromes for the ring errors in T_c. A detected error is announced if $S(I)$ does not appear in the table. After F has been found, the result I is corrected to the value $I \ominus F$. At this point, we leave it to the reader to imagine whether such error correction could ever be practically implemented, but we shall give further attention to implementation in Sections 4 and 6.

2.5. Modular Weight and Modular Distance

For results I of arithmetic operations in Z_m, we mentioned in Section 2.4 that if

$$I = AN + E = AN \oplus F$$

then the actual error E is either F or $F - m$ depending on whether $E \geq 0$ or $E < 0$. Hence, the arithmetic weight of the actual error is either $W(F)$ or $W(F - m) = W(m - F) = W(\ominus F)$. Since $W(E)$ has already been justified as an appropriate measure of adder malfunction for modulo m addition with $m = 2^n - 1$ or $m = 2^n$, we are led naturally to the following measure of the weight of ring errors proposed by Rao and García ([10]):

Definition 2.3. The modular weight of an integer F in Z_m is the minimum of $W(F)$ and $W(\ominus F)$, and is denoted $W_m(F)$.

It follows immediately that the modular weight of F cannot exceed the arithmetic weight of F.

We note first that modular weight, for any m, satisfies the first two norm properties, namely

$$W_m(F) = W_m(\ominus F) \tag{17}$$

and

$$W_m(F) \geq 0 \quad \text{with equality if and only if} \quad F = 0 \tag{18}$$

Unfortunately, the triangle inequality

$$W_m(F_1 \oplus F_2) \leq W_m(F_1) + W_m(F_2) \tag{19}$$

does not hold for arbitrary m. For instance, with $m = 35$, $F_1 = 3$, and $F_2 = 19$, one has

$$W_{35}(3 \oplus 19) = W_{35}(22) = 3 > W_{35}(3) + W_{35}(19) = 1 + 1 = 2$$

Fortunately, (19) does hold for the cases of greatest practical interest, namely $m = 2^n - 1$ and $m = 2^n$, as we now show in the following theorem due to García ([5]). Because of the great importance which we shall attach to this theorem in the following development, we include a complete proof even though the proof is quite lengthy. The proof also lends insight into why (19) fails for an arbitrary modulus m.

Theorem 2.4. For any integers F_1 and F_2 in Z_m, the triangle inequality (19) for modular weight is satisfied if m is any of the moduli $2^n - 1$, 2^n, or $2^n + 1$.

Proof. Suppose first that $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(\ominus F_2)$ If also $F_1 + F_2 < m$, we have

$$W_m(F_1 \oplus F_2) = W_m(F_1 + F_2) \leq W(m - F_1 - F_2)$$

and thus by the triangle inequality (5) for arithmetic weight,

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(m - F_2) = W_m(F_1) + W_m(F_2)$$

so the theorem is proved for this case. Similarly, if $F_1 + F_2 \geq m$,

$$W_m(F_1 \oplus F_2) = W_m(F_1 + F_2 - m) \leq W(F_1 + F_2 - m)$$

so that again by (5) we have

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(m - F_2) = W_m(F_1) + W_m(F_2)$$

This completes the proof for $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(\ominus F_2)$, and by symmetry for $W_m(F_1) = W(\ominus F_1)$ and $W_m(F_2) = W(F_2)$. We note that we have not yet made use of our hypothesis on the form of m.

Suppose, momentarily, that the theorem holds when $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(F_2)$. Then the theorem also holds for $W_m(F_1) = W(\ominus F_1)$ and $W_m(F_2) = W(\ominus F_2)$, since then, by property (17),

$$\begin{aligned} W_m(F_1 \oplus F_2) &= W_m[(\ominus F_1) \oplus (\ominus F_2)] \leq W_m(\ominus F_1) + W_m(\ominus F_2) \\ &= W_m(F_1) + W_m(F_2) \end{aligned}$$

Hence, the theorem will be proved if we can prove it for the case where $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(F_2)$, as we now proceed to do.

If also $F_1 + F_2 < m$, we have

$$W_m(F_1 \oplus F_2) = W_m(F_1 + F_2) \leq W(F_1 + F_2)$$

and again by (5) we have

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(F_2) = W_m(F_1) + W_m(F_2)$$

If $F_1 + F_2 = m$, we have $W_m(F_1 \oplus F_2) = 0$, so the theorem holds trivially.

It remains to consider the case $F_1 + F_2 > m$. We now impose our restriction on m and suppose first that $m = 2^n - 1$. By our hypotheses, $m = 2^n - 1 < F_1 + F_2 < 2m < 2^{n+1}$. It follows from Corollary 1.1 that the NAF of $F_1 + F_2$ is given by the $(n + 2)$-tuple $[b_{n+1}, b_n, \ldots, b_1, b_0]$, where either (b_{n+1}, b_n) is $(0, 1)$ or $(1, 0)$. In the former case, the term $+2^n$ appears in the NAF of $F_1 + F_2$, and hence

$$W(F_1 + F_2 - 2^n) = W(F_1 + F_2) - 1$$

Thus,

$$\begin{aligned} W_m(F_1 \oplus F_2) &= W_m(F_1 + F_2 - m) \leq W(F_1 + F_2 - m) \\ &= W(F_1 + F_2 - 2^n + 1) \end{aligned}$$

can be overbounded using the triangle inequality (5) as

$$W_m(F_1 \oplus F_2) \leq W(F_1 + F_2 - 2^n) + W(1) = W(F_1 + F_2)$$

and another application of (5) gives

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(F_2) = W_m(F_1) + W_m(F_2)$$

Finally, we must consider the case when $(b_{n+1}, b_n) = (1, 0)$, i.e., when

$+2^{n+1}$ is a term in the NAF for $F_1 + F_2$. Repeating the same steps as above, we find

$$W(F_1 + F_2 - 2m) \leq W(F_1 + F_2) \leq W_m(F_1) + W_m(F_2)$$

But also from (17) we have

$$W_m(F_1 \oplus F_2) = W_m[(\ominus F_1) \oplus (\ominus F_2)] = W_m[(m - F_1) \oplus (m - F_2)]$$

Noting that $0 < 2m - F_1 - F_2 < m$, it follows that

$$\begin{aligned} W_m(F_1 \oplus F_2) &= W_m(2m - F_1 - F_2) \leq W(2m - F_1 - F_2) \\ &= W(F_1 + F_2 - 2m) \end{aligned}$$

where we made use of (3) in the last step. It follows that again

$$W_m(F_1 \oplus F_2) \leq W_m(F_1) + W_m(F_2)$$

so that the theorem is proved for the modulus $m = 2^n - 1$. We leave to the reader the task of making the minor changes in the proof required for the moduli 2^n and $2^n + 1$.

It follows from the norm properties (17)–(19) that for the moduli $2^n - 1$, 2^n, and $2^n + 1$, we can use modular weight to define a distance function for Z_m just as we used arithmetic weight to define a distance function for any set of integers.

Definition 2.4. The *modular distance* between the integers I_1 and I_2 in Z_m, where m is one of the moduli $2^n - 1$, 2^n, or $2^n + 1$, is the modular weight of their difference and will be denoted $D_m(I_1, I_2)$, i.e.,

$$D_m(I_1, I_2) = W_m(I_1 \ominus I_2) \tag{20}$$

It follows immediately from (17)–(19) that modular distance is a true metric for Z_m satisfying the following three defining properties of a metric:

$$D_m(I_1, I_2) = D_m(I_2, I_1) \tag{21}$$

$$D_m(I_1, I_2) \geq 0 \quad \text{with equality if and only if} \quad I_1 = I_2 \tag{22}$$

$$D_m(I_1, I_2) \leq D_m(I_1, I_3) + D_m(I_3, I_2) \tag{23}$$

For any of the moduli $2^n - 1$, 2^n, or $2^n + 1$, the *minimum modular distance* of an AN-code with $m = AB$ is defined as the minimum of the modular distances between pairs of distinct code points in the code. But

we note that if AN_1 and AN_2 are distinct code points, then

$$D_m(AN_1, AN_2) = W_m[A(N_1 - N_2)] = \min[W(AN_3), W(AN_4)]$$

where without loss of generality we have supposed $N_1 > N_2$ and where we define $N_3 = N_1 - N_2$ and $N_4 = B - N_3$. It follows that $0 < N_3 < B$ and $0 < N_4 < B$, which shows that the modular distance between distinct code points is always the arithmetic weight of some nonzero code point. Hence, the minimum modular distance cannot be smaller than $W_{\min}$, the minimum arithmetic weight of the nonzero code points. Conversely, if AN_5 is a code point of weight $W_{\min}$, then $D_m(0, AN_5) \leq W(AN_5) = W_{\min}$, so that the minimum modular distance cannot exceed $W_{\min}$. It follows that the minimum modular distance is exactly $W_{\min}$ and hence by Theorem 2.1, exactly equal to the minimum arithmetic distance of the code. We summarize as the following theorem.

Theorem 2.5. For an AN-code where $m = AB$ is one of the moduli $2^n - 1$, 2^n, or $2^n + 1$, the minimum modular distance and the minimum arithmetic distance coincide. (Hence, the same symbol $D_{\min}$ will be used for both of these minimum distances.)

We now prove the following fundamental result for modular distance, which is the analog of Theorem 2.2 for arithmetic distance.

Theorem 2.6. For any $t \geq 0$ and any $s \geq 0$, an AN-code where $m = AB$ is one of the moduli $2^n - 1$, 2^n, or $2^n + 1$ can correct all ring errors of modular weight t or less in its code points and can detect all other ring errors of modular weight $t + s$ or less in its code points if and only if its minimum modular distance satisfies

$$D_{\min} > 2t + s$$

Proof. Suppose that $D_{\min} > 2t + s$. We shall show that the decoder which decodes the result I into the nearest code point AN_1 if there is a code point AN_1 with $D_m(I, AN_1) \leq t$, and announces a detected error otherwise, performs the claimed correction and detection. For, suppose $I = AN \oplus F$, where AN is the correct result and $W_m(F) \leq t$. Then, $D_m(I, AN) \leq t$, but for any other code point AN', we have by the triangle inequality (23) that $D_m(I, AN') \geq D_m(AN, AN') - D_m(I, AN) > t + s$, so that $N_1 = N$ and the error is corrected as claimed. Similarly, if $t < W_m(F) \leq t + s$, then for any other code point AN', $D_m(I, AN') > t$, so that no code point within distance t of I exists and a detected error will be announced as claimed.