The syndrome coding technique and its application to fast event selection in high-energy physics experiments

N. M. Nikityuk
Joint Institute for Nuclear Research, Dubna

Fiz. Elem. Chastits At. Yadra **24**, 180–245 (January–February 1993)

The problems in the use of the theory and practice of algebraic coding theory for data compression and processing in multichannel charged-particle detectors are discussed. A system of analogies between the theories of correcting codes and hodoscopic systems is described. The main result of this approach is the development of radically new logic units, such as parallel encoders for $t>1$ signals, majority coincidence circuits for a large number of inputs, dynamically programmable modules, and special-purpose processors with algebraic structure based on elements of the Galois field $GF(2^m)$ for fast event selection. It is also shown how calculations in Galois fields can be used to design dynamically program-controlled logic modules, which appear promising for use in trigger systems.

INTRODUCTION

In spite of their simple mathematical structure, Galois fields hold great promise for use in information technology and in other areas of science and technology, for example, in electronics methods in high-energy physics. In Fig. 1 we show the areas in which finite-field algebras are applied. More than 30 years ago, Peterson[1] applied Galois field theory for decoding codes correcting $t>1$ errors. This area has continued to develop up to the present time. In turn, algebraic coding theory has stimulated the development of three new areas in contemporary information technology: source encoding,[2] signature analysis[3] for designing systems for testing microprocessors and large-scale integrated circuits, and the syndrome coding technique,[4] proposed by the author for fast data compression and the development of parallel encoders for fast coding of $t>1$ signals using combinational circuits.[5,6] Such encoders are unique. A number of theorems and ideas of algebraic coding theory have been used by the author to design majority coincidence schemes for a large number of inputs ($n>100$), special-purpose processors for fast event selection in spectrometers used in high-energy physics, and time-to-number converters, and parallel encoders for light coding of devices for compressing and processing data recorded in two-coordinate detectors.[7–9] This method has also been used to construct a trigger processor.[10]

There are also other areas where Galois fields are used: in Fourier transformation,[11] and in sequential automatons, where the inputs and outputs are encoded by Galois field elements, which play an important role in computational techniques[12–16] and in nuclear electronics.[9]

Here we give a systematic review of the studies on the development and use of the syndrome coding technique for compression and analysis of data recorded in multichannel charged-particle detectors. The method is based on the use of algebraic coding theory, which, in turn, is based on the methods of the algebra of Galois fields.

OPERATIONS ON ELEMENTS OF THE GALOIS FIELD

In contrast to the ordinary, familiar binary arithmetic, in the syndrome coding technique arithmetic and algebraic operations are carried out on an extended Galois field $GF(p^m)$, where p is a prime number, called the characteristic of the field. For a binary system $p=2$ and m is an integer. The rules for performing operations on the field elements are described in several studies.[17–19] To facilitate the reading of this review by readers unfamiliar with the methods of performing arithmetic and algebraic operations on elements of the Galois field $GF(2^m)$, here we give the necessary information.

The elements of a Galois field can be added, multiplied, inverted, and so on, just like the ordinary, more familiar rational, real, and complex numbers. The basic difference is that the latter contain infinitely many elements, whereas the Galois field, by definition, contains only a finite number of elements[13] $n=p^m$. A field containing p^m elements is denoted by $GF(2^m)$. Consequently, the Galois field $GF(2^1)$ contains the two elements 0 and 1. The number of nonzero elements of a finite field is equal to the degree of its characteristic, i.e., $n=2^m-1$, where the number of different elements of the field is referred to as its order. All the elements of a given field can be obtained simply by using an irreducible polynomial. Tables of such polynomials up to the 34th degree are given in Appendix C of Ref. 1. In what follows, to construct various schemes for the compression and analysis of data obtained using multichannel charged-particle detectors containing $n=15$, 31, and 63 channels we shall use irreducible polynomials of degree 4, 5, and 6: X^4+X+1, X^5+X^2+X+1, and X^6+X+1. It should be noted that in our examples the $+$ sign denotes mod-2 summation.

Among the field elements there are m linearly independent (basis) elements. For example, for the polynomial $f(X)=X^6+X+1$ ($m=6$) we have the following basis elements: $a^0=100000$, $a^1=010000$, $a^2=001000$, $a^3=000100$, $a^4=000010$, and $a^5=000001$. One of them, the element a^1,

1063-7796/93/010077-30$10.00

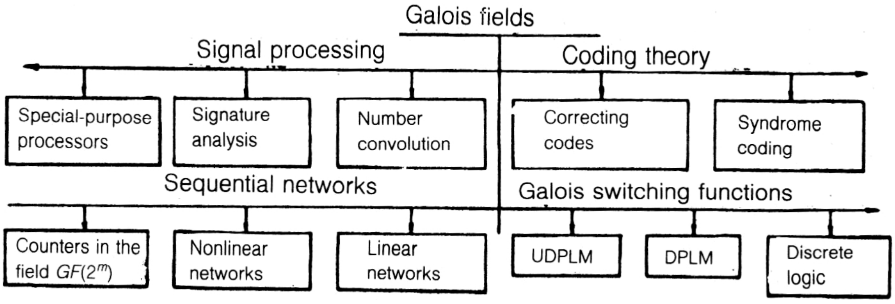

FIG. 1. Applications of Galois algebra in various areas of science and technology. UDPLM stands for Universal Dynamically Programmable Logic Module. DPLM stands for Dynamically Programmable Logic Module.

is a root of the polynomial $f(X)$. Therefore, each nonzero field element can be represented as a power of the element a^1. In other words, the multiplicative group of a finite field is cyclical in nature. Furthermore, the smallest positive number for which $a^n=a^0=1$ is called the order of the element a^1. If the order of the element a^1 is equal to n, the elements a^0, a^1, a^2,...,a^{n-1} are distinct. Thus, in our example $n=2^m-1=63$. Therefore, for example, $a^{126}=(a^{63})^2=a^0$, $a^{64}=a^{63}$, $a^1=a^1$, and so on. Using the fact that a^1 is a root of the polynomial $f(X)$, the other elements of the field $GF(2^6)$ can be obtained from the relation $a^6+a^1+1=0$, i.e., $a^6=a^1+1=110000$, $a^7=a^2+a^1=011000$, and so on. A list of all the elements of this field is given in the Appendix.

It is often convenient to express a field element A as a polynomial of degree $m-1$ (a vector): $A=d_0a^0+d_1a^1+d_2a^2+...+d_{m-2}a^{m-2}+d_{m-1}a^{m-1}$, where the coefficients d_0, d_1, d_2,...,d_{m-2}, d_{m-1} are 0 or 1. For example, for the element $a^9=a^4+a^3$, $d_0=d_1=d_2=d_5$ and $d_3=d_4=1$. Just as in ordinary arithmetic, in a Galois field there are differences in carrying out operations on the field elements by computer and by hand. This is primarily true of such operations as multiplication, division, raising to a power, and extracting the square root. The operation of multiplication is simple to do by hand: the degree of the product is equal to the sum of the degrees of the factors (taking into account the cyclic nature of the field). The operation of division of an element A by an element B is equivalent to multiplying the element A by the inverse element B^{-1}. In turn, the inverse B^{-1} of the element B is found from the expression $BB^{-1}=1=a^0$. For example, for the element a^9 the element a^{54} is the inverse element, since $a^9a^{54}=a^{63}=a^0$. A simple rule can be used for calculations done by hand: the degree of the inverse element having degree k is 2^m-1-k. For example, the degree of the inverse element a^{25} is 38. Next let us consider extracting the square root. If i is an even number, then $(a^i)^{1/2}=a^{i/2}$. If i is an odd number, the degree of the element corresponding to $(a^i)^{1/2}$ is $(i+2^m-1)/2$. For example, $(a^5)^{1/2}=a^{34}$. The operation of raising field elements to a power is performed as for ordinary numbers, except that it is performed modulo n. For example, $(a^{60})^{12}=a^{720}=a^{63\times 11}a^{27}=a^{27}$. Such operations as addition and subtraction in the field $GF(2^m)$ are equivalent and are performed modulo two.

HARDWARE REALIZATION OF CERTAIN OPERATIONS IN A GALOIS FIELD $GF(2^m)$

Counters

To obtain a sequence of field elements in both the forward and the backward directions, shift registers with logical feedback are used.[1] The structure of the connections in such circuits depends on the irreducible polynomial chosen for constructing the field. In Fig. 2 we show the registers in a Galois field for counting in the forward and backward directions modulo the polynomial X^6+X+1. If the lowest bit of the register shown in Fig. 2a is set to 1 and the others are set to 0, successive shifts of the register give a representation of successive powers of the element a^1 in the form in which they are given in the Appendix. The presence of feedback from the highest bit to the lowest one makes it possible to obtain the value $a^6=a^1+1$. Shifting to the left (Fig. 2b) corresponds to division by a^1 such that 1 carried from the lowest bit gives the value $a=1+a^5$ (Ref. 6).

Multiplication and raising to a power

The multiplication of two field elements for a given basis can be performed if these elements are represented as polynomials.[19] For example, if one field element A is written as

$$A=a^0a_0+a^1a_1+a^2a_2+a^3a_3+a^4a_4+a^5a_5,$$

and another element as

$$B=a^0b_0+a^1b_1+a^2b_2+a^3b_3+a^4b_4+a^5b_5,$$

the direct multiplication of these polynomials mod 6 gives the product of two elements of the field $GF(2^6)$, which is

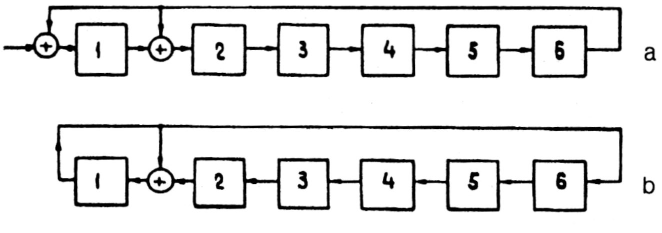

FIG. 2. Counters in a Galois field modulo the polynomial X^6+X+1: (a) counter in the forward direction; (b) counter in the backward direction; 1–6 are the shift-register elements, and ⊕ are mod-2 adders.

simple to realize by using AND logic elements and a parity-checking scheme. Such circuits are essentially multi-input mod-2 adders. Denoting the corresponding coordinates of the product element by g_0, g_1, g_2, g_3, g_4, and g_5, we obtain the following Boolean expressions for the product of two elements:

$$\begin{aligned}
g_0&=a_0b_0+a_1b_5+a_2b_4+a_3b_3+a_4b_2+a_5b_1,\\
g_1&=a_2b_4+a_2b_5+a_3b_3+a_3b_4+a_4b_2+a_5b_2+a_0b_1+a_1b_0\\
&\quad+a_1b_5+a_4b_3+a_5b_1,\\
g_2&=a_0b_2+a_1b_1+a_3b_4+a_3b_5+a_4b_3+a_4b_4+a_5b_2+a_5b_3\\
&\quad+a_2b_0+a_2b_5,\\
g_3&=a_0b_3+a_1b_2+a_3b_0+a_3b_5+a_4b_4+a_4b_5+a_5b_4+a_5b_3\\
&\quad+a_2b_1, \qquad\qquad (1)\\
g_4&=a_0b_4+a_1b_3+a_2b_2+a_3b_1+a_4b_0+a_4b_5+a_5b_4+a_5b_5,\\
g_5&=a_0b_5+a_1b_4+a_2b_3+a_3b_2+a_4b_1+a_5b_0+a_5b_5.
\end{aligned}$$

If in Eq. (1) we set $A=B$, we obtain the Boolean expressions for the field element (we shall assume it is element B) squared:

$$\begin{aligned}
&g_0=b_0+b_3, \quad g_1=b_3, \quad g_2=b_1+b_4,\\
&g_3=b_4, \quad g_4=b_2+b_5, \quad \text{and } g_5=b_5.
\end{aligned}\tag{2}$$

By computer iteration it is easy to obtain the Boolean expressions for any field element raised to a higher power.

Extraction of the square root

The extraction of the square root and solution of a second-degree equation in a field of characteristic 2 are linear operations, and so their hardware realization is fairly simple. For example, we have the relations $(X+Y)^2=X^2+Y^2$ and $(X+Y)^{1/2}=X^{1/2}+Y^{1/2}$ (Ref. 20). To extract the square root of the element A it is necessary to transform from one basis a^0, a^1, a^2, a^3, a^4, a^5 to another basis $(a^0)^{1/2}=a^0$; $(a^1)^{1/2}=a^{32}$; $(a^2)^{1/2}=a^1$; $(a^3)^{1/2}=a^{33}$; $(a^4)^{1/2}=a^2$; $(a^5)^{1/2}=a^{34}$. Then the expression for extracting the square root of the element B is obtained from the matrix equation

$$|b_0,b_1,b_2,b_3,b_4,b_5|\times\begin{vmatrix}a^0\\a^{32}\\a^1\\a^{33}\\a^2\\a^{34}\end{vmatrix}=A^{1/2}.$$

For example, $(a^{25})^{1/2}=a^{44}$ or

$$[010001]\times\begin{vmatrix}1&0&0&0&0&0\\1&0&0&1&0&0\\0&1&0&0&0&0\\0&1&0&0&1&0\\0&0&1&0&0&0\\0&0&1&0&0&1\end{vmatrix}=a^{44}.$$

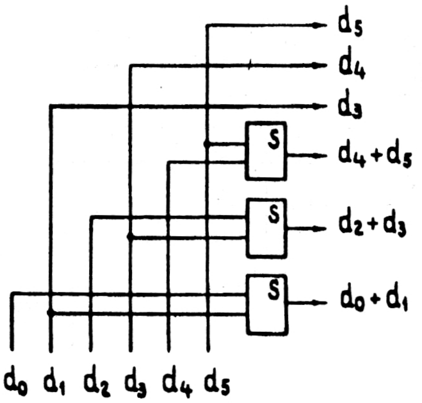

FIG. 3. Basic circuit for finding the square of an arbitrary element in the field $GF(2^6)$. Here S is the SN7486 chip.

As an example, in Fig. 3 we show the basic circuit for computing the square root of any element of the field $GF(2^6)$. Methods of solving equations in a Galois field will be discussed below.

Calculation of the inverse element

To calculate the inverse B^{-1} of the element B it is necessary to raise the latter to the power 2^m-2, since $BB^{2m-2}=B^{2^{m-1}}=1$. It is even simpler to calculate the inverse element by using a programmable memory, i.e., by tabular arithmetic techniques.

GALOIS SWITCHING FUNCTIONS

A Galois switching function over a field $GF(p^n)$ can (which is very important) be represented as a polynomial of degree less than p^n (Refs. 12 and 13). This representation is widely used in the theory of Galois switching functions (GSFs). Since the Galois field $GF(2^m)$ is a natural extension of a Boolean field, the representation of switching functions as polynomials where both the variables and the coefficients are field elements has a number of fundamental advantages.

1. It is possible to perform algebraic operations on GSFs, which simplifies the problem of minimization and its formal representation.

2. In a Galois field there are operations like addition, multiplication, and division which give additional advantages over Boolean fields.[16]

3. The representation of switching functions in the form of polynomials makes it possible to use standard programming systems to calculate complicated logic structures.

4. Since the input and output states of a logic structure are coded by field elements, the following state can be represented as a polynomial function of the current state and the current output.

5. The description of multivalued and multilevel schemes is very compact, which simplifies their theoretical study.

TABLE I. Elements of the field $GF(2^3)$ (a^2+a+1).

Elements	Decomposition in basis elements	Binary equivalents			Minterms
0	0	000	$GF(2)$		$\bar{X}_0\bar{X}_1\bar{X}_2$
a^0	1	100			$X_0\bar{X}_1\bar{X}_2$
a^1	a^1	010			$\bar{X}_0X_1\bar{X}_2$
a^2	a^2	001			$\bar{X}_0\bar{X}_1X_2$
a^3	$1 \quad +a^1$	110		$GF(2^3)$	$X_0X_1\bar{X}_2$
a^4	$a^1 \quad +a^2$	011			$\bar{X}_0X_1X_2$
a^5	$1 \quad +a^1 \quad +a^2$	111			$X_0X_1X_2$
a^6	$1 \quad +a^2$	101			$X_0\bar{X}_1X_2$
$a^7=a^0$	1	100			$X_0\bar{X}_1\bar{X}_2$

Below, we shall present the latest results obtained by the author on the practical use of GSFs for constructing fast programmable modules and processor devices for electronics methods in high-energy physics experiments. In addition, we shall consider the fast algorithms which have been developed and devices for performing combined operations on elements of a Galois field $GF(2^m)$. For this we have used analytic calculations and computer calculations using such programming languages as PL/1, REDUCE, and SCOONSCHIP.[21–23] Readers interested in more details about the properties of GSFs may consult the literature.[24–28]

As an example, in Tables I and II we give three possible ways of representing the field elements of $GF(2^3)$ and $GF(2^4)$.

Let us consider some examples of using GSFs. It is well known that any GSF $F(X)=F(X_0,X_1,X_2,...,X_{m-1})$ of m arguments in the field $GF(2^m)$ can be represented as a polynomial:[13]

$$F(X)=B(0)+A(1)X+A(2)X^2+A(3)X^3+...+A(2^m-1)^{2^{m-1}}. \qquad (3)$$

The coefficients $A(k)$ are calculated from the expression

TABLE II. Elements of the field $GF(2^4)$ (a^4+a+1).

Elements	Expansion in basis elements	Binary equivalents			Minterms
0	0	0000	$GF(2)$		$\bar{X}_0\bar{X}_1\bar{X}_2\bar{X}_3$
a^0	1	1000			$X_0\bar{X}_1\bar{X}_2\bar{X}_3$
a^1	a^1	0100			$\bar{X}_0X_1\bar{X}_2\bar{X}_3$
a^2	a^2	0010			$\bar{X}_0\bar{X}_1X_2\bar{X}_3$
a^3	a^3	0001			$\bar{X}_0\bar{X}_1\bar{X}_2X_3$
a^4	$1 \quad +a^1$	1100			$X_0X_1\bar{X}_2\bar{X}_3$
a^5	$a^1 \quad +a^2$	0110			$\bar{X}_0X_1X_2\bar{X}_3$
a^6	$a^2 \quad +a^3$	0011			$\bar{X}_0\bar{X}_1X_2X_3$
a^7	$1 \quad +a^1 \quad +a^3$	1101		$GF(2^4)$	$X_0X_1\bar{X}_2X_3$
a^8	$1 \quad +a^2$	1010			$X_0\bar{X}_1X_2\bar{X}_3$
a^9	$a^1 \quad +a^3$	0101			$\bar{X}_0X_1\bar{X}_2X_3$
a^{10}	$1 \quad +a^1 \quad +a^2$	1110			$X_0X_1X_2\bar{X}_3$
a^{11}	$a^1 \quad +a^2 \quad +a^3$	0111			$\bar{X}_0X_1X_2X_3$
a^{12}	$1 \quad +a^1 \quad +a^2 \quad +a^3$	1111			$X_0X_1X_2X_3$
a^{13}	$1 \quad +a^2 \quad +a^3$	1011			$X_0\bar{X}_1X_2X_3$
a^{14}	$1 \quad +a^3$	1001			$X_0\bar{X}_1\bar{X}_2X_3$
a^{15}	1	1000			$X_0\bar{X}_1\bar{X}_2\bar{X}_3$

TABLE III.

Inputs $X=\{X_0, X_1, X_2\}$	Outputs $B(a_j)$	
$0=000$	$0=000=0$	0
$a^0=100$	$a^0=100=1$	0
$a^1=010$	$a^0=100=1$	0
$a^2=001$	$a^0=100=1$	0
$a^3=110$	$a^1=010=0$	1
$a^4=011$	$a^1=010=0$	1
$a^5=111$	$a^3=110=1$	1
$a^6=101$	$a^1=010=0$	1
$a^7=100=a^0$	$a^0=100=1$	0
	↓ S	↓ C

$$A(k)=\sum_{i=1}^{2^m-1} a_i^{-k}[B(0)+B(a_j)],\quad k=1,2,3,\ldots,2^{m-1},$$

where $B(a_j)$ are substitution elements obtained from the table of inputs and outputs, and $B(0)$ is the value of the function at zero. Therefore, the following sequence of steps for synthesizing GSF variables can be proposed:

1. An irreducible polynomial of degree m is selected from the tables of Ref. 1, and all the nonzero elements of the field $GF(2^m)$ are found. For large values of m these calculations can be done by computer.

2. The input–output correspondence table is constructed.

3. The coefficients $A(k)$ are calculated.

4. The polynomial representation of the function is expanded in basis elements, and similar terms are combined, taking into account the fact that the addition is mod-2.

Example 1 (Refs. 21 and 22). Let us consider GSFs over the field $GF(2^3)$ formed by using the irreducible polynomial $P_3=X^3+X+1$. We assume that $a^0=100$, $a^1=010$, and $a^2=001$ are linearly independent field elements, and the element a^1 is a root. Under these conditions it is easy to obtain the other four elements: $a^4=a^3a^1=a^2+a^1=011$, $a^5=a^4a^1=a^3+a^2=a^2+a^1+a^0=111$, $a^6=a^5a^1=a^2=a^0=101$, and $a^7=a^0$. Suppose that we want to construct a scheme for a one-bit complete adder. Let us consider the input–output correspondence table (Table III).

The quantities X_0, X_1, and X_2 correspond to the first and second bits of the sum and the carry, respectively, and S and C denote the sum and carry. It follows from Table III that the elements of the substitution $B(1)\rightarrow B(7)$ in our example are the elements of the field $GF(2^3)$.

For methodological purposes let us consider in more detail the process of calculating the coefficients $A(k)$:

$$A(1)=\frac{a^0}{a^0}+\frac{a^0}{a^1}+\frac{a^0}{a^2}+\frac{a^1}{a^3}+\frac{a^1}{a^4}+\frac{a^3}{a^5}+\frac{a^1}{a^6}$$
$$=a^0+a^0a^6+a^0a^5+a^1a^4+a^1a^3+a^3a^2+a^1a^1=a^0.$$

The operation of division is replaced by multiplication by the inverse element. Then we have

$$A(2)=\frac{a^0}{(a^0)^2}+\frac{a^0}{(a^1)^2}+\frac{a^0}{(a^2)^2}+\frac{a^1}{(a^3)^2}+\frac{a^1}{(a^4)^2}+\frac{a^3}{(a^5)^2}$$
$$+\frac{a^1}{(a^6)^2}=a^1.$$

Similar calculations give $A(3)=a^0$, $A(4)=A(7)=0$, $A(5)=a^4$, $A(6)=101$. Then Eq. (3) has the form

$$F(X)=X^1+a^1X^2+a^4X^5+a^6X^6+X^3. \tag{4}$$

Current technology permits two realizations of Eq. (4). 1) Tabular arithmetic methods can be used. However, this would be too awkward. 2) Equation (4) can be expanded in basis elements in order to obtain Boolean expressions. For this it is sufficient to represent the coefficients a^4, a^6 and the values of the variable X^1, X^2, X^3, X^5, and X^6 as polynomials. For example, $a^4=a^1+a^2$, $a^6=a^0+a^2$, $X^2=X_0+X_2a^1+(X_1+X_2)a^2$, and so on. We have

$$F(X)=(X_0+X_1a^1+X_2a^2)+a^1[X_0+X_2a^1$$
$$+(X_1+X_2)a^2]+[(X_0+X_1+X_2+X_1X_2)]$$
$$+[(X_1+X_0X_1+X_0X_2)a^1+(X_2+X_0X_1)a^2]$$
$$+(a^1+a^2)(X_0+X_1+X_2+X_1X_2)$$
$$+(X_1+X_2+X_0X_2)a^1+(X_1+X_0X_1+X_0X_2)a^2$$
$$+(a^0+a^2)[(X_0+X_1+X_2+X_1X_2)$$
$$+(X_2+X_0X_1)a^1+(X_1+X_2X_0)a^2].$$

After multiplication and collection of similar terms, we obtain the Boolean expressions describing the operation of a one-bit adder in the logic AND and exclusive-OR logic base:

$$S=X_0+X_1+X_2\quad\langle a^0\rangle$$

$$C=X_0X_1+X_0X_2+X_1X_2\ \langle a^1\rangle.$$

Example 2. Let us design a circuit for a sequential automaton coded by the correspondence table given below (Table IV). In other words, elements of the field $GF(2^4)$ are supplied to the automaton inputs in the order of in-

TABLE IV.

Inputs	Outputs
$X=\{X_0, X_1, X_2, X_3\}$	$F(X)$
$0 = 0000$	0
$a^0 = 1000$	a^1
$a^1 = 0100$	0
$a^2 = 0010$	a^7
$a^3 = 0001$	a^5
$a^4 = 1100$	a^{10}
$a^5 = 0110$	a^{11}
$a^6 = 0011$	a^{13}
$a^7 = 1101$	a^0
$a^8 = 1010$	a^3
$a^9 = 0101$	a^{14}
$a^{10} = 1110$	a^3
$a^{11} = 0111$	0
$a^{12} = 1111$	a^8
$a^{13} = 1011$	a^4
$a^{14} = 1001$	a^0

creasing degree. These elements are easily generated by using a register in the field $GF(2^4)$. The same elements are obtained at the automaton outputs, but in the order given in Table IV. For constructing the automaton circuit it is necessary to calculate the 16 coefficients of the polynomial

$$\begin{aligned}F(X_0,X_1,X_2X_3)=B(0)&+A(1)X+A(2)X^2+A(3)X^3\\&+A(4)X^4+A(5)X^5+A(6)X^6\\&+A(7)X^7+A(8)X^8+A(9)X^9\\&+A(10)X^{10}+A(11)X^{11}\\&+A(12)X^{12}+A(13)X^{13}\\&+A(14)X^{14}+A(15)X^{15}.\qquad(5)\end{aligned}$$

The coefficients $A(1)$–$A(15)$, like the expansion in basis elements and collection of similar terms, were done by computer. The following Boolean expressions were obtained:

$$\begin{aligned}&X_0X_1+X_2+X_0X_2+X_0X_3+X_1X_2+X_1X_3+X_0X_1X_3\\&+X_1X_2X_3+X_0X_1X_2X_3\quad\langle a^0\rangle\qquad(6)\\&X_0+X_2+X_3+X_1X_3+X_0X_1X_3+X_1X_2X_3\quad\langle a^1\rangle\\&X_3+X_0X_1+X_0X_3+X_1X_2+X_1X_3+X_1X_2X_3\\&+X_0X_1X_2X_3\quad\langle a^2\rangle\\&X_2+X_1X_3+X_0X_1X_3+X_0X_2X_3\quad\langle a^3\rangle.\end{aligned}$$

In Fig. 4 we show the basic automaton circuit, and in Fig. 5 we show the basic circuit of the modulo summator con-

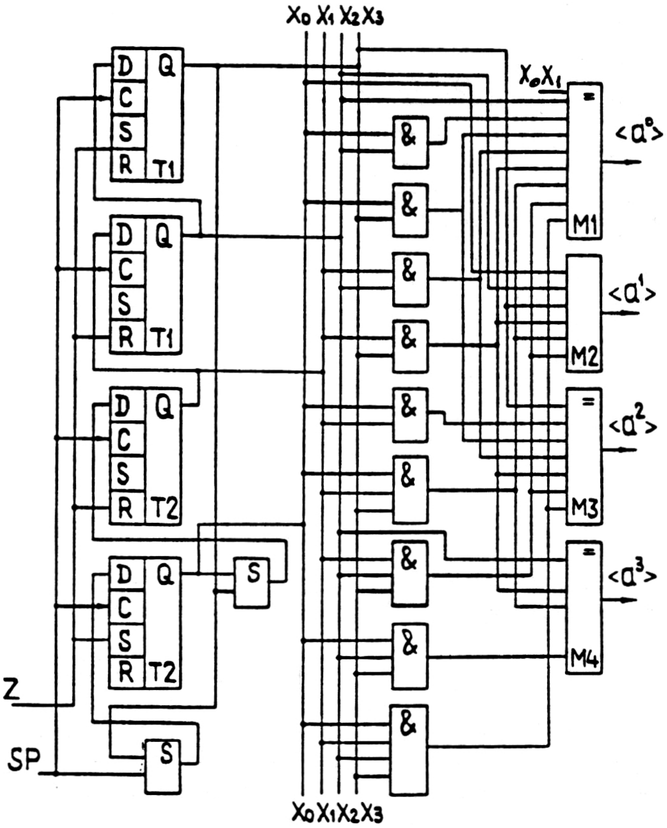

FIG. 4. Basic circuit for a sequential automaton: $T1$, $T2$, and D are triggers, S is a mod-2 adder, and M1–M4 are SN74180 parity-checking circuits.

taining 144 inputs. Such circuits together with AND logic elements are needed for the fast execution of various operations in the field $GF(2^m)$.

Example 3. Let us design a circuit for a three-bit adder with carry. This device must contain six inputs and four outputs. This means that the calculations must be performed with GSFs of six variables in the field $GF(2^6)$. The elements of this field are treated as 6-bit binary numbers X_0, X_1, X_2, X_3, X_4, and X_5. Of these, the first three digits represent the first term, and the last three represent the second. The following Boolean expressions were obtained by computer:

$$\begin{aligned}&X_0X_3+X_0X_1X_3+X_1X_3X_4+X_0X_1X_2X_5+X_0X_2X_4X_5\\&\quad+X_1X_2X_3X_5+X_2X_3X_4X_5\qquad\langle a^0\rangle\rightarrow\text{1st bit}\\&X_0+X_3+X_0X_4+X_1X_2X_5+X_2X_4X_5\quad\langle a^1\rangle\rightarrow\text{2nd bit}\\&X_1+X_4+X_2X_5\qquad\langle a^2\rangle\rightarrow\text{3rd bit}\\&X_2+X_5\qquad\langle a^3\rangle\rightarrow\text{carry.}\end{aligned}$$

In Fig. 6 we show the basic circuit of the adder. It should be noted that it is difficult to construct by hand the input–output correspondence table for this number of variables. In our example this was done by computer, as were the calculations.[28]

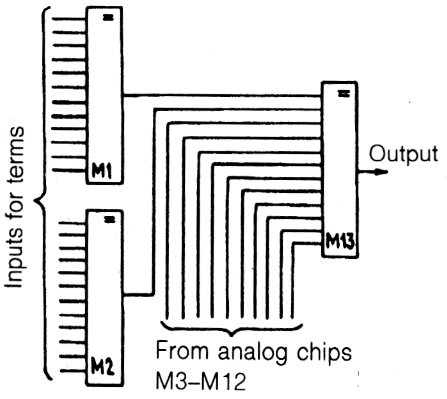

FIG. 5. Basic circuit of the mod-2 adder with 144 inputs: M1–M13 are MC10160 chips.

A →

B ↓

x	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}
a^0	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}
a^1	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0
a^2	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1
a^3	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2
a^4	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3
a^5	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4
a^6	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5
a^7	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6
a^8	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7
a^9	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8
a^{10}	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9
a^{11}	a^{11}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
a^{12}	a^{12}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}
a^{13}	a^{13}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}
a^{14}	a^{14}	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}

FIG. 7. Multiplication table for two elements in the field $GF(2^4)$.

COMBINED (JOINT) OPERATIONS AND THEIR APPLICATION

Analysis of Eqs. (3) and (4) shows that the hardware realization of GSFs together with such operations as addition, multiplication, division, and raising to a power in the field $GF(2^m)$ require the calculation of complicated expressions, like the simultaneous multiplication (division) of terms raised to powers. The problem is how the features of Galois fields can be used to find effective algorithms for the hardware realization of complicated algebraic expressions. There have been several studies devoted to parallel methods of performing operations in a Galois field (Refs. 14, 19, 29, and 31). In turn, the author has proposed several fast algorithms, including the technique of combined operations.[30,31] We shall use the term "combined operation" to mean simultaneous calculation without the use of clock pulses containing several elements raised to powers. There are three ways of carrying out combined operations.

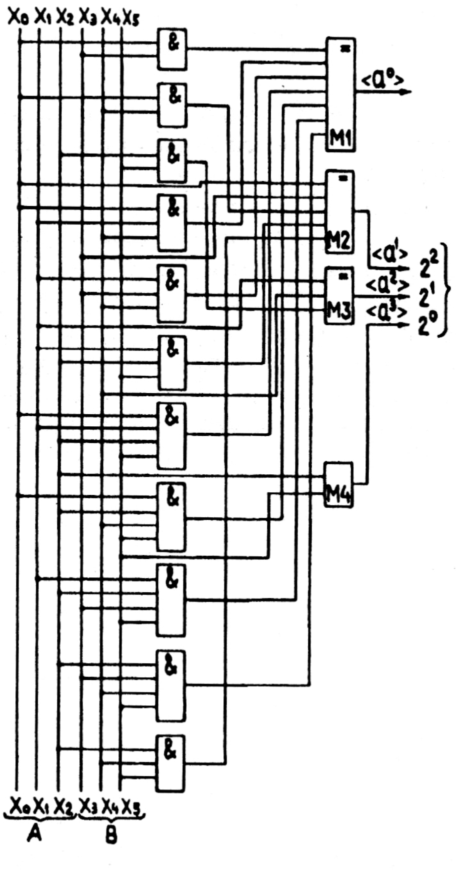
X_0 X_1 X_2 X_3 X_4 X_5
&
&
&
&
&
&
&
&
&
&
&
M1
M2
M3
M4
$<a^0>$
$<a^1>$
$<a^2>$
$<a^3>$
2^2
2^1
2^0
X_0 X_1 X_2 X_3 X_4 X_5
A
B

FIG. 6. Basic circuit of a three-bit adder: M1–M4 are SN74180 or MC10160 chips, and & is the AND logic element.

1. The tabular method

The use of this method is based on the simple fact that no matter how complex an expression is, the value of one of the field elements is obtained as the result. From the practical point of view this means that the number of address inputs of the programmable memory is determined

A →

↓ B

BA^3	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}
a^0	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}
a^1	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}
a^2	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}
a^3	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0
a^4	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1
a^5	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2
a^6	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3
a^7	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4
a^8	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5
a^9	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6
a^{10}	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7
a^{11}	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8
a^{12}	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9	a^{12}	a^0	a^3	a^6	a^9
a^{13}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}	a^{13}	a^1	a^4	a^7	a^{10}
a^{14}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}	a^{14}	a^2	a^5	a^8	a^{11}

FIG. 8. Table for the multiplication of element B by element A^3.

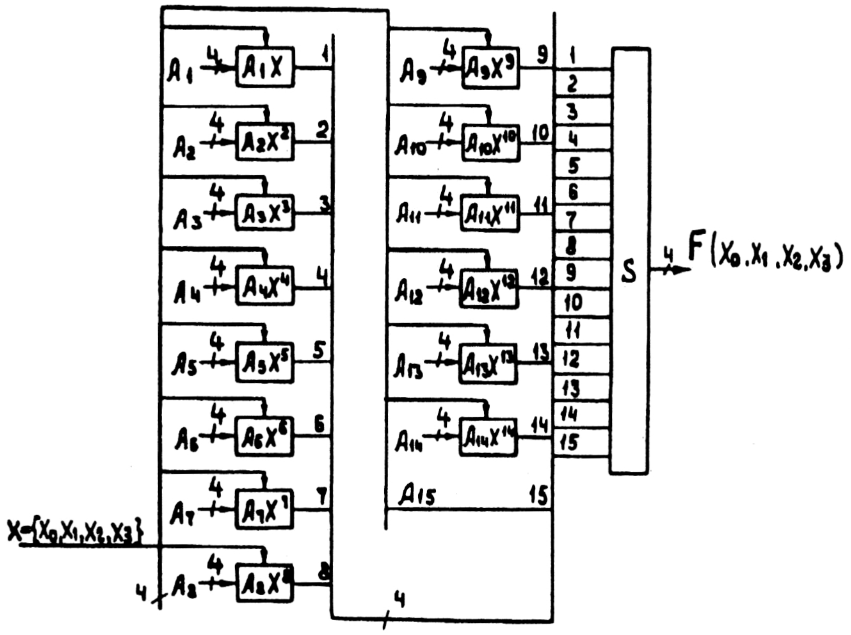

FIG. 9. Diagram of a 4-variable UDPLM: S is a parity-checking circuit.

only by the number of elements entering into the given expression. For example, the size of the memory containing the solution table will be the same in the realization of the following expressions:

$$F_1=\frac{ABC+D}{DE} \quad \text{and}$$

$$F_2=A^pB^qC^r(D^s)^{-1}(E^t)^{-1}+D+C,$$

where A, B, C, D, E, F_1, and F_2 are elements of the field $GF(2^m)$, and p, q, r, s, and t are integers or fractions. It is easily seen that a memory with $5m$ inputs is required for calculating F_1 and F_2. Let us consider an example of the use of the tabular method to perform combined operations.

Synthesis of a universal dynamically programmable logic module. This area is currently receiving a great deal of both theoretical and, particularly, practical attention. The problem is basically that the criteria for selecting physical events are becoming more complicated, making fast switching of trigger systems essential in many new problems.[29] The goal is to design a logic device which will perform the maximum number of logic operations for the minimum number of control inputs (tuning coefficients) as economically and quickly as possible. It follows from Eq. (5) that:

1) The coefficients $A(1)$–$A(15)$ determine the type of GSF that is realized.

2) The calculation of a polynomial altogether requires carrying out operations on field elements such as addition, multiplication, and raising to a power.

By using the technique for carrying out combined operations, it is possible to considerably simplify the calculation of Eq. (5). In Fig. 7 we give the multiplication table for two elements A and B in the field $GF(2^4)$. In Fig. 8 we give the analogous table but with element B raised to the third power. For example, the expression BA^3 for $A=a^7$ and $B=a^{12}$ is equal to a^3. Or, in greater detail, $a^{12}(a^7)^3=a^{12}a^{21}=(a^{33})=a^{15}a^{15}a^3=a^3$. In this manner it is possible to write down 14 tables which can be used to program the contents of the memory module. In Fig. 9 we show the circuit of an UDPLM of four variables. It contains 4 inputs for the variables, 60 inputs for the tuning coefficients, and 4 outputs.[29] We see from this figure that the construction of such a module requires 15 circuits for simultaneous multiplication with raising to a power and a mod-2 adder. A fast MC10149 memory module can be used for this. In Fig. 10 we show a circuit for carrying out combined operations. It consists of a 4-bit register in which the value of the tuning coefficient is stored and an MC10149 chip. By varying the coefficients $A(k)$, where $k=1, 2, 3,...,15$, it is possible to tune the UDPLM to perform any of 65536 switching functions of four variables.[29,30]

Let us consider several of the simplest logic functions shown in Table V. In the first column on the left we give the elements of the field $GF(2^4)$ and their binary equivalents. The signals corresponding to these codes are fed to the 4 inputs of the module. In the other columns we give the values corresponding to them obtained at the module outputs and the coefficients calculated by computer. We assume that the GSF has its true value if it is equal to the unit element $a^0=1000$. The following coefficients are obtained for 4-fold coincidences: $A(1)=a^3$, $A(2)=a^6$,

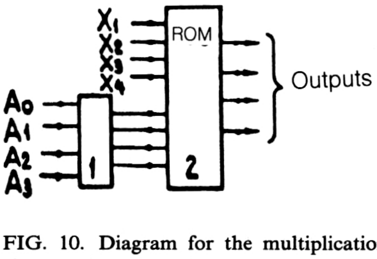

FIG. 10. Diagram for the multiplication of element B by the power element A: 1 is a register, and 2 is the MC10149 programmable memory module.

TABLE V. Representation of some logic functions of elements of the field $GF(2^4)$.

Inputs/function	Coincidence		Repeat		Inversion	
$X=X_0,X_1,X_2,X_3$	$f(X)=X_0,X_1,X_2,X_3$	$A(k)$	$f(X) = X$	$A(k)$	$f(X)=\bar{X}_0,\bar{X}_1,\bar{X}_2,\bar{X}_3$	$A(k)$
$a^1 = 0100$	0000	a^3	a^1	a^0	1011	a^0
$a^2 = 0010$	0000	a^6	a^2	0000	1101	0000
$a^3 = 0001$	0000	a^9	a^3	0000	1110	0000
$a^4 = 1100$	0000	a^{12}	a^4	0000	0011	0000
$a^5 = 0110$	0000	a^0	a^5	0000	1001	0000
$a^6 = 0011$	0000	a^3	a^6	0000	1100	0000
$a^7 = 1101$	0000	a^6	a^7	0000	0010	0000
$a^8 = 1010$	0000	a^9	a^8	0000	0101	0000
$a^9 = 0101$	0000	a^{12}	a^9	0000	1010	0000
$a^{10} = 1110$	0000	a^0	a^{10}	0000	0001	0000
$a^{11} = 0111$	0000	a^3	a^{11}	0000	1000	0000
$a^{12} = 1111$	1000	a^6	a^{12}	0000	0000	0000
$a^{13} = 1011$	0000	a^9	a^{13}	0000	0100	0000
$a^{14} = 1001$	0000	a^{12}	a^{14}	0000	0110	0000
$a^{15} = a^0 = 1000$	0000	a^0	a^{15}	0000	0000	a^{12}

$A(3)=a^9,...,A(14)=a^{12}$, and $A(15)=a^0$. As a result we obtain the equation

$$F(X)=a^3X+a^6X^2+a^9X^3+a^{12}X^4+a^0X^5+a^3X^6$$
$$+a^6X^7+a^9X^8+a^{12}X^9+a^0X^{10}+a^3X^{11}$$
$$+a^6X^{12}+a^9X^{13}+a^{12}X^{14}+a^0X^{15}. \qquad (7)$$

The fact that the use of Eq. (7) describes a coincidence circuit with four inputs can be checked by two methods.

1. Substituting $X=a^{12}$, we obtain $a^3a^{12}=a^0$, $a^6(a^{12})^2=a^{30}=a^{15}a^{15}=a^0$, and so on: 15 identical values a^0. After mod-2 addition we obtain $F(X)=a^0$. If into Eq. (7) instead of the variable X we successively substitute the other values of the elements, then each time in Eq. (7) we obtain 15 different elements a^0-a^{14}, the sum of which is equal to zero by definition. It is easy to see that it makes no sense to design a coincidence circuit in this form.

2. If Eq. (7) is simplified by expanding the field elements in the basis elements, after simplification by computer we obtain the following equation:

$$F(X)=F(X_0X_1X_2X_3)=a^0.$$

This means that the UDPLM outputs take the value a^0, i.e., the true value, if all the inputs are in the logical 1 state.

Let us consider the second column in Table V, which illustrates the tuning of the module for performing a "passive" operation, when the values at the inputs and outputs are identical (repeat of logic symbols). In this case only one coefficient is nonzero. We therefore have

$$F(X)=a^0X=a^0(a^0X_0+a^1X_1+a^2X_2+a^3X_3)=X.$$

If we substitute the corresponding coefficients a^0 of X and a^{12} of X^{15} into Eq. (5), we obtain the logic equation for inversion, except for zero:

$$F(X)=a^0X+a^{12}X^{15}=X+a^{12}.$$

2. A fast algorithm for performing combined operations

The realization of this algorithm requires AND logic elements and mod-2 adders. We shall describe the method by means of examples. First let us consider the operation of multiplying two elements A and B in the field $GF(2^4)$. We have

$$P=AB=(A_0a^0+A_1a^1+A_2a^2+A_3a^3)$$
$$\times(B_0a^0+B_1a^1+B_2^2a+B_3^3a)$$
$$=P_0a^0+P_1a^1+P_2a^2+P_3a^3,$$

where

$$P_0=A_0B_0+A_1B_3+A_2B_2+A_3B_1 \qquad \langle a^0\rangle$$
$$P_1=A_0B_1+A_1B_0+A_1B_3+A_2B_2+A_2B_3+A_3B_2$$
$$+A_3B_1 \qquad \langle a^1\rangle$$
$$P_2=A_0B_2+A_1B_1+A_2B_0+A_2B_3+A_3B_2+A_3B_3 \qquad \langle a^2\rangle$$
$$P_3=A_0B_3+A_1B_2+A_2B_1+A_3B_0+A_3B_3 \qquad \langle a^3\rangle.$$

Setting $A=B$, we obtain the Boolean expressions for squaring any element A:

$$P_0^2=A_0+A_2 \qquad \langle a^0\rangle$$
$$P_1^2=A_2 \qquad \langle a^1\rangle$$
$$P_2^2=A_1+A_3 \qquad \langle a^2\rangle$$
$$P_3^3=A_3 \qquad \langle a^3\rangle.$$

Obviously, to obtain the expressions describing the combined operation of multiplication of B by A^2, we need to perform the following calculations:

$$BA^2=(B_0a^0+B_1a^1+B_2a^2+B_3a^3)$$
$$\times[(A_0+A_2)a^0+A_2a^1+(A_1+A_3)a^2+A_3a^3].$$

After simplification we obtain

$$K_0=B_0A_0+B_0A_2+B_2A_1+B_3A_2+B_1A_3+B_2A_3 \quad \langle a^0\rangle$$
$$K_1=B_0A_2+B_1A_2+B_2A_1+B_1A_3+B_3A_2+B_3A_1$$
$$+B_3A_3+B_1A_0 \quad \langle a^1\rangle$$
$$K_2=B_0A_1+B_0A_3+B_1A_2+B_2A_0+B_2A_2+B_3A_1 \quad \langle a^2\rangle$$
$$K_3=B_0A_3+B_1A_1+B_1A_3+B_2A_2+B_3A_0+B_3A_2$$
$$+B_3A_3 \quad \langle a^3\rangle.$$

Example. Let $B=a^3$ and $A=a^{12}$. Then

$$BA^2=a^3(a^{12})^2=a^{27}=a^{12},$$

$$B_3=A_0=A_1=A_2=A_3=1, \quad \text{and} \quad B_0=B_1=B_2=0.$$

We find

$$K_0=K_2=K_1=K_3=1 \quad \text{and} \quad BA^2=a^3(a^{12})^2=a^{12}.$$

To realize combined operations on the basis of combinational logic circuits, the author has proposed a module performing such procedures as $B^kA^k(B^k/A^k)$, where $k=1-2^{m-2}$. For $m=4$ the module consists of a matrix of AND logic elements, 14 groups of mod-2 adders, and a multiplexer which switches the operation of the module to perform one of the operations of the type BA^2, B^2A, B^2A^3, and so on (Fig. 11).[30] Such an approach can also be used to construct modules performing more complicated functions.

3. Use of logarithms and antilogarithms

For the efficient realization of complicated expressions in a Galois field, Berlekamp[14] proposed that the logarithms and antilogarithms of elements of the field $GF(2^m)$ on the base of a^1 be used. For example, in the Galois field $GF(2^4)$ the degree of the product of two elements $A=a^7$ and $B=a^{10}$ is equal to two. In fact, $\log_a a^7=0111_2$ and $\log_a a^{10}=1010_2$ (lowest bit on the left). Adding the resulting binary numbers, we obtain 0010_2. Similar calculations can be performed when an element A is divided by an element B. Logarithms and antilogarithms are easily obtained by tabular arithmetic methods using programmable memory modules, as was shown in Ref. 42. By using the simple rules for logarithms, we can perform the addition operation by replacing it by multiplication:

$$a^i+a^j=a^i(1+a^{i-j}).$$

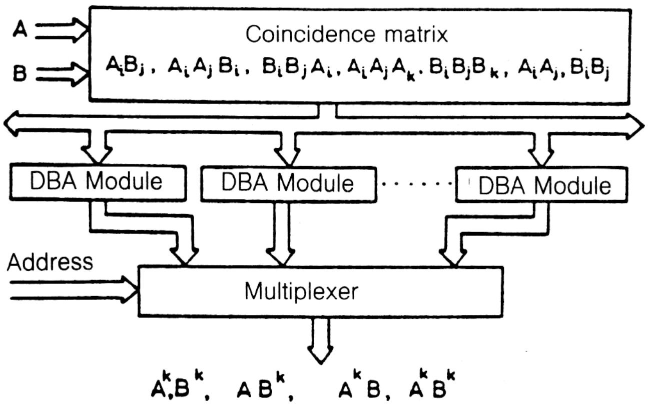
A
B
Coincidence matrix
A_iB_j, $A_iA_jB_i$, $B_iB_jA_i$, $A_iA_jA_k$, $B_iB_jB_k$, A_iA_j, B_iB_j
DBA Module
DBA Module
DBA Module
Address
Multiplexer
$A^k_{\cdot}B^k$, AB^k, A^kB, A^kB^k

FIG. 11. Version of a chip for performing combined operations on two elements A and B in the field $GF(2^4)$.

To shorten the time for the cyclical summation of powers of field elements when there are many terms, the author has proposed a method of cyclical compression and a device corresponding to it. In Fig. 12 we show two examples illustrating the algorithm for the operation of a cyclical compressor. The first example on the left corresponds to the simultaneous multiplication of 15 elements a^0-a^{14} or, equivalently, the raising of the element a^0 to the 15th power. The (mod 15) cyclical summation process in this example can conveniently be divided into five stages. In the first stage as a result of counting the number of 1s in each column the 15 terms are compressed to four. The result of the summation is written along the diagonal beginning with the first column on the right. A similar procedure is carried out in the second and third stages of the summation. In the end, the 15 terms are reduced to two, split into two parts. The second part of the sum is the largest number (11010000), which is equal to the sum of the carrys arising in the cyclical compression process. Finally, to the number 0010 we add 1101 mod 15. In Fig. 12 on the right we give an example for the simultaneous calculation of the sum of powers of the product $a^{10}a^{14}a^9a^8a^7a^6a^5=a^{14}$ in the field $GF(2^4)$. These examples serve simultaneously as the diagrams for constructing the basic circuit of a cyclical compressor (Fig. 13). In this scheme we do not show the programmable or read-only memory (ROM) modules used to calculate the logarithms and antilogarithms. We see from Fig. 13 that the first cascade of the compressor consists of four (15,4) parallel registers which essentially form a tree of complete adders.[32] The second cascade of the group of parallel registers has a smaller number of inputs. After the third stage of compression only two terms are obtained, which are added by using an ordinary adder with cyclical carry. High operating speed of the cyclical compressor is obtained by the use of parallel registers.

Let us summarize our conclusions. By using GSFs and analytic computer calculations it is possible to synthesize complex logic devices, including programmable modules, which are very promising for use in first- and second-level trigger systems. In the following sections we shall show how algebraic coding theory based on the theory of the Galois field is used for fast compression and analysis of data in high-energy physics experiments.

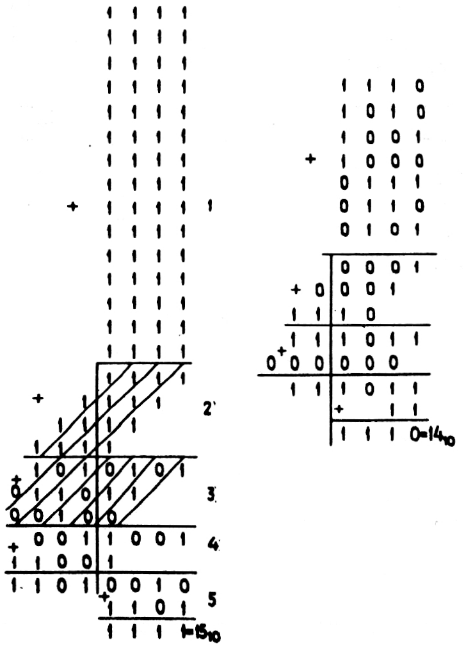

FIG. 12. Two examples illustrating the operation of a parallel cyclical compressor in the field $GF(2^4)$.

THE SYNDROME CODING TECHNIQUE

As is well known, modern multichannel charged-particle detectors contain an enormous number of position-sensitive detectors (communication channels). Fortunately, only some fraction (10–20%) of these detectors is triggered during the recording of an event. The signals from these detectors are amplified, shaped, and processed using fast electronics and specialized processors for the purpose of generating a fast signal triggering a physical device. The selection (filtration) of useful events is multilevel in nature. In terms of time, the first-level decision must be made within 100 ns and occasionally less. The main event-selection criteria at the first level are the multiplicity t of the recorded events, the types of particle, the particle emission angle, the presence of a decay vertex, and so on. At later levels it is possible to use more complicated criteria such as determination of the event coordinates under conditions of high multiplicity, calculation of the particle momenta, calculation of the magnitude of the energy release using calorimeters, and so on.

The operation of a typical multilevel trigger system is illustrated in Table VI (Ref. 33). Measurement of the time begins at the first picosecond, when the signal formation begins in the detectors, and runs up to the time at which the candidate for a useful event is written on magnetic tape.

In Fig. 14 we show a simple scheme consisting of a target T and a detector. We assume arbitrarily that the detector consists of 31 elements, for example, scintillators located in a single plane. The passage of two particles can give rise to two or more simultaneous signals. After amplification and shaping, they are fed to the inputs of a majority coincidence scheme (MCS) for determining the signal multiplicity t and to the inputs of an encoder used to determine the particle coordinates (Fig. 15). A single particle can often trigger two or more scintillators. In such cases (as in calorimeters) the problem of cluster recording and identification arises. If it is known for certain that $t=1$, then the construction of a parallel encoder using only combinational circuits (without memory elements) is not difficult. Shift registers and priority encoders are often used to solve the problem of $t>1$. However, the operation of such circuits requires clock pulses, so that when the number of recording channels n is large a great deal of time is required to determine the event coordinates. Modern programmable read-only memory devices (PROMs) or programmable logic matrices (PLMs) have a limited number of inputs (of order 15–30) for variables, so that their use cannot solve the problem of fast coding at large $n>30$. The use of the syndrome coding method makes it possible to solve the problem of constructing a fast MCS and a parallel encoder with a large number of channels for $t>1$ (Ref. 34).

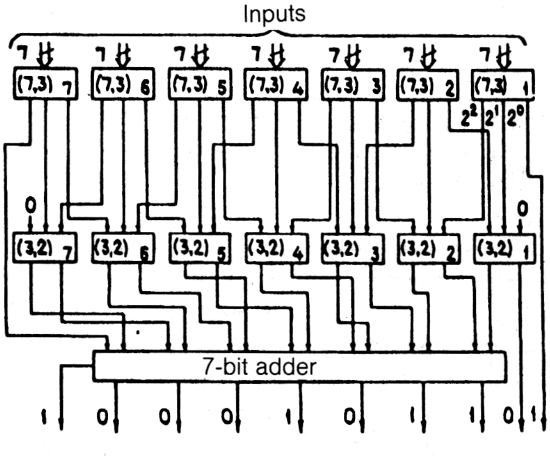

FIG. 13. Circuit for a compressor in the field $GF(2^4)$.

A SYSTEM OF ANALOGIES

To apply algebraic coding theory to hodoscopic systems, it is useful to consider a system of analogies existing between the basic conclusions and concepts used in algebraic coding theory and algebraic methods of signal processing in hodoscopic systems. In Table VII we give such a system of analogies.[34]

Let us comment on Table VII line by line.

1. Redundant coding is used in communications and computational technology to increase the reliability of communications devices and information processing. Redundant coding in this sense is widely used in electronics methods for high-energy physics experiments. However, studies by the author have shown that the technique of correcting codes can be used successfully also for designing detectors and hodoscopic systems with the optimal relation between the multiplicity of recorded signals, the spatial resolution, and simplicity of the coding scheme.

TABLE VI. Diagram illustrating the event-analysis process in fixed-target and collider experiments.

Fixed target		Colliding beams	
	1 ps		1 ps
	10 ps		10 ps
	100 ps		100 ps
First level			
	1 ns	←Particles pass through silicon detector→	1 ns
		←Writing of all signals	
	10 ns		
		Particles leave the large detector→	
Writing of all the data	100 ns		100 ns
		Period of beam intersection. Writing of all the signals→	
	1 μs		1 μs
Second level			
Data transfer to event processor	10 μs	Calculation of p_t^*	10 μs
	100 μs	Selection of useful tracks and data on decay vertices	100 μs
		Calculation of invariant particle masses	
	1 ms	Start of track event reconstruction	1 ms
Third level	10 ms		10 ms
	100 ms	←Event written to magnetic tape	100 ms
		Event written to magnetic tape→	1 s

p_t^* is the particle momentum in the calorimeter.

2. A block code is a correcting code in which a sequence of n symbols is used. The code vector of a block code consists of k information and $m=n-k$ extra control bits. A vector consisting only of zeros is called a null vector. A null vector corresponds to a null word read out from the detector or the hodoscopic plane in the case where no data elements are triggered. In contrast to coding theory, where a code word is usually treated as an ordinary binary code transmitted via a communication channel, in hodoscopic systems the information is read from position-sensitive detectors, and so the coordinates of the triggered detectors are obtained in the form of a unitary position code.

3. An error vector e can be added to the code vector during the data transfer along a channel. In the theory of hodoscopic systems this vector corresponds to a physical event, as a result of which signals are fed from the detectors via the recording channels.

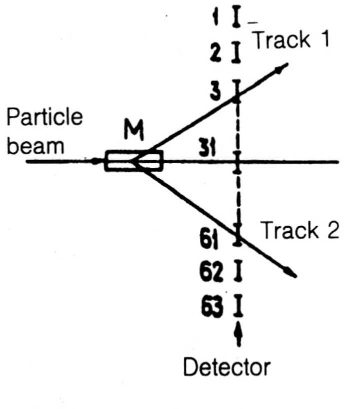

FIG. 14. Simple scheme illustrating the method of recording two particles in a scintillation hodoscope: M is the target; 1–63 are scintillators. A single plane is shown.

4. An error burst in transmission channels corresponds to a cluster which arises as a result of the triggering of a group of adjacent detectors. Therefore, the theory of error-burst correcting codes can be used to construct devices capable of recording cluster coordinates.

5. The parameter t is the number of distorted information symbols which can be corrected by a given code. In the theory of hodoscopic systems the value of t determines the maximum number of triggered detectors, the coordinates of which are uniquely determined by the coordinate processor.

6. An important code parameter is the number of check symbols (the code syndrome). This quantity depends on the construction of the code, the block length, and the parameter t. For example, if we take the widely known Hamming code, for which $t=1$, the block length is $n=2^m-1$ and the number of information symbols is $k=n-m$. If $t>1$, for optimal codes the number of check symbols is $N=mt$.

7. The code efficiency v is determined from the ratio $v=k/n$. In hodoscopic systems this parameter corresponds

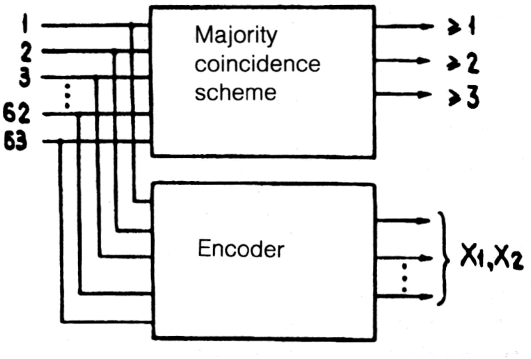

FIG. 15. Circuit for fast event selection.

TABLE VII.

Algebraic coding theory	Algebraic theory of signal processing in hodoscopic systems
1. Redundant coding	Redundant coding for the purpose of increasing the spatial resolution and increasing the functional possibilities of hodoscopes
2. Code vector of a block code consisting of n symbols	Code word computed from n position-sensitive detectors
3. Error vector e	Physical event recorded in a multichannel detector
4. Error burst	Event with a cluster
5. Correcting capability of the code t	Number of simultaneously triggered position-sensitive detectors t
6. Number of check symbols mt	Number of bits at the outputs of a parallel encoder
7. Code efficiency	Compression coefficient K_c
8. Coding device	Parallel encoder
9. Check matrix	Coding matrix
10. Code distance d	Code distance d
11. Weight of the code vector	Weight of a row of the coding matrix
12. Weight of a column of the check matrix	Coefficient for splitting of signal from an element, K_p
13. A completely asymmetric channel	Recording channel in the hodoscopic plane
14. Iterated coding	Coding of data in a detector consisting of several hodoscopic planes

to compression coefficient K_c equal to n/N. This is one of the most important parameters of a hodoscope using syndrome coding, since it characterizes the degree of data compression.

8. A coding device forms the control symbols at the transmitter end. In hodoscopic systems the analog of a coding device is a parallel encoder. In scintillation hodoscopes the coding can be performed at the level of the hodoscope construction, for example, by installing the corresponding connections between the scintillators and PMs using light guides. It should be noted that the coding device can operate either in parallel or in sequential fashion using shift registers with logic feedback.

9. The structure of the coding device is specified by the matrix of check ratios consisting of zeros and ones and containing N rows and n columns. Examples of such matrices will be encountered frequently in this review. In accordance with the analogy, parallel encoders with specified properties can be constructed by using check matrices which, when employed in hodoscopic systems, will be referred to as coupling matrices or coding matrices. Coupling matrices can be used to find not only the basic parameters of the parallel encoders, but also their basic circuits. In order to eliminate the uncertainties in calculating the coordinates of the triggered detectors or even one of them, all the columns of the coupling matrix should be different, and each position-sensitive element should correspond to a single column. Linear and nonlinear operations are performed on the columns of the corresponding coupling matrix in order to analyze the functional possibilities of a parallel encoder. For example, if all possible sums of two columns are different, such a matrix can be used to construct a parallel encoder for determining the coordinates of at least two triggered position-sensitive detectors. In addition, the number of rows in the coupling matrix corresponds to the number of outputs N of the parallel encoder.

10. One of the most important parameters of a correcting code is its code distance d. Knowing the code distance of a given code, it is possible to select or construct the coding matrix with the required properties. Since the initial code word read out from the detector is treated as a null word, it is often more convenient to use the following theorem:[65] a linear (n,k) code with check matrix $H=[h_0,h_1,h_{n-1}]$, where h_i, $i=0,1,\ldots,n-1$ are column vectors of dimension $[n-k]\times 1$, has the minimum code distance d if and only if any $d-1$ columns of the coding matrix $H_{n,N}$ are linearly independent. With this theorem we can calculate coding matrices with given properties by computer, especially for large values of n.

11. The weight w of a code vector is defined as the number of nonzero components of this vector. The number of ones in a line of the check matrix, as in a line of the coupling matrix, characterizes how difficult it is to realize coding devices, since the number of inputs of the parity-checking scheme is equal to the number of ones in the rows of the coding matrix.

12. The weight of a column of the check matrix is also related to the complexity of realizing a coding device or a parallel encoder. The weight of a column of the coupling matrix determines the signal branching coefficient K_p. The smaller this quantity with the other parameters fixed, the simpler the coupling between the detector outputs and the logic elements or amplifiers of the parallel encoder.

13. A channel in which only one type of error occurs is completely asymmetric, i.e., either transformation of zeros into ones or only ones into zeros is possible. In this sense the data-readout channels in hodoscopic systems are purely asymmetric. The need to introduce such an analogy is dictated by the fact that several codes with good parameters are available for asymmetric channels, and such codes are easy to realize.

14. The two-dimensional iterated code used for optimal data coding both in scintillation hodoscopes and in MPWCs is the simplest iterated code. In this coding method the n detectors are arranged as a square matrix containing k rows and k columns, and the syndrome for the rows and the columns is calculated. As a result, the number of signal amplifier-mixers (PMs) is decreased to $2n^{1/2}$. The syndrome is calculated by using an amplifier-mixer. It should be noted that in a two-dimensional iterated code the total code distance is $d=d_1d_2$, where d_1 and d_2 are the code distances of the initial codes selected for

iteration. It is important that iterated codes form an extensive class of codes which is interesting from the practical point of view, since any codes with good initial parameters can be chosen for iteration.

We have considered the main analogs and, in general, this list could be continued. For example, important practical conclusions can be made on the basis of the following analogy. In coding theory there is a so-called superposition code which is rarely used in practice. It contains M code words such that for a positive number g the Boolean sum of g different code words differs from each sum of g or fewer code words. In addition, these sums must differ from the terms. For example,

$$\begin{array}{cccccc} & 1000 & & 1101 & & 0110 \\ \vee & & \vee & & \vee & \\ & \underline{0100} & & \underline{0011} & & \underline{0001} \\ & 1100 & & 1111 & & 0111 \end{array}$$

In practice, this means that encoders for superposition codes can be designed on the basis of signal mixers (PMs), ordinary amplifier-mixers, fiber-optics light guides, and so on. However, when all the other parameters are fixed, superposition codes have a smaller code distance and, accordingly, smaller compression coefficient K_c, since the syndrome formation is performed not modulo two, but according to the rules for a Boolean sum, where the number of code combinations obtained is smaller. For example, $1+1=0$, $1+1+1+1=0$, $1+0=1$, $0+1=1$ (mod 2). Meanwhile, by the rules for Boolean addition we have $1+1=1$, $1+1+1+1=1$, $0+1=1$, $1+0=1$.

Thus, the essence of the syndrome coding technique is the following (Fig. 16). First let us consider a typical multibit transmission system using correcting codes (Fig. 16a), which contains a one-bit register on the transmitter side. In the process of transmission and coding to a one-bit word a k-bit syndrome code is added in accordance with the selected code. Then the $(1+k)$-bit word is transmitted to a device containing the decoding apparatus. If errors arose during the transmission via the communication channel, they are corrected (within certain limits) and then the one-bit word is stored in the register. Now let us consider a simpler transmission scheme used in an experiment (Fig. 16b). In the absence of an event or for false triggering of position-sensitive detectors, the information word is always equal to zero, and the event detection is treated as the addition of an error vector to a null information word, the length of which is equal to the number of recording channels n. If the optimal code correcting t errors is used, at the output of the parallel encoder the length of the readout word is compressed to $N=\log_2 n$. For example, assume that the number of recording channels is 31, and that signals are sent from 10 and 22 transmission channels:

0000000001000000000001000000000

(the channels are counted from left to right). If the BCH (Bose–Chaudhuri–Hocquenghem) code correcting $t=2$ errors is used, at the output of the encoder the following syndrome N is obtained:

$$N=\log_2 31=10.$$

We therefore arrive at the following conclusions.

For $t \ll n$ (this condition is usually satisfied in practice) the efficiency of the syndrome coding technique increases with increasing n. When a BCH code is used at the encoder outputs, a code one obtains which is equivalent to a certain set of elements of the Galois field $GF(2^m)$ on which various arithmetic and algebraic operations can be performed.

The syndrome code carries information both about the number of triggered position-sensitive detectors and about their coordinates.

Thus, the use of the syndrome coding technique involves the following procedures: 1) data compression by means of parallel encoders t; 2) calculation of the signal

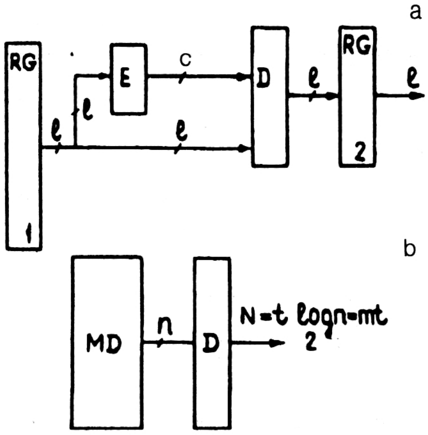

FIG. 16. Explanation of the syndrome coding technique. (a) An ordinary transmission system, where correcting codes are used; C is an encoder, and D is a decoder. (b) Multichannel transmission system, in which the syndrome coding technique is used; MD is a multichannel detector.

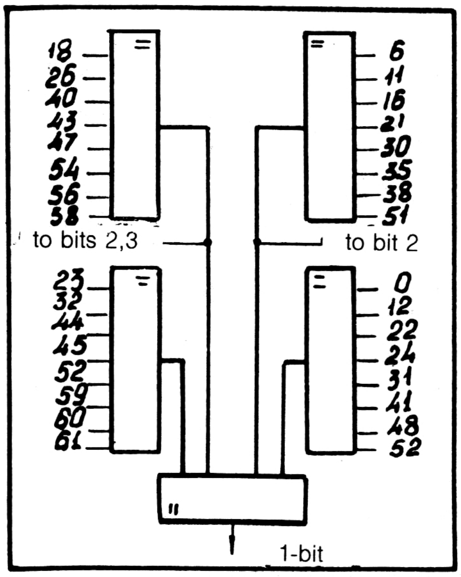

FIG. 17. Basic circuit for calculating one bit of a syndrome, using MC10160 chips.

multiplicity; 3) determination of the coordinates of the triggered position-sensitive detectors (data elements) in the hodoscopic plane of the detector.

PARALLEL ENCODERS

In accordance with the syndrome coding technique, the data elements of the detector are numbered in order of increasing degree of the elements of the field $GF(2^m)$. The transformation of the unitary code read out from a multichannel detector into field elements after amplification and formation of the signals is done by means of an encoder which, as usual in the theory of BCH codes, is described by means of a matrix of check ratios[34–36] H^T (coding matrix $H_{n,mt}$). A detailed description of BCH code theory can be found in Ref. 1. The general form of the coding matrix $H_{n,mt}$ is

$$H_{n,mt}=\left|\begin{array}{cccc} 1 & 1 & 1 & 1 \\ a^1 & a^3 & a^5 & a^{2t-1} \\ a^2 & a^6 & a^{10} & a^{2(2t-1)} \\ a^3 & a^9 & a^{15} & a^{3(2t-1)} \\ \cdots & \cdots & \cdots & \cdots \\ a^{n-1} & a^{3(n-1)} & a^{5(n-1)} & a^{(n-1)(2t-1)} \end{array}\right| \qquad (8)$$

We see that the first column contains the elements of the field $GF(2^m)$ in the order of increasing degree. The second column contains the cubes of the corresponding elements in the first column, and so on. To simplify matters we assume that the multichannel detector has $n=2^6-1=63$ data elements ($m=6$). This means that the Galois field is formed by means of the polynomial X^6+X+1. For multiplicity $t\leqslant 4$ the coding matrix $H_{63,24}$ has the form

$$\begin{array}{rc|cccc|c|cccc|} \text{Positions of} & & & & & & & & & & \\ \text{data elements} & & & & & & & \multicolumn{4}{c}{\text{Binary equivalents}} \\ 0^{*} & & 1 & 1 & 1 & 1 & & 100000 & 100000 & 100000 & 100000 \\ 1 & & a^1 & a^3 & a^5 & a^7 & & 010000 & 000100 & 000001 & 001100 \\ 2 & & a^2 & a^6 & a^{10} & a^{14} & & 001000 & 110000 & 000011 & 001010 \\ 3^{*} & & a^3 & a^9 & a^{15} & a^{21} & & 000100 & 000110 & 000101 & 110111 \\ 4 & H_{63,24}= & a^4 & a^{12} & a^{20} & a^{28} & \longrightarrow & 000010 & 101000 & 001111 & 001110 \\ 5^{*} & & a^5 & a^{15} & a^{25} & a^{35} & & 000001 & 000101 & 010001 & 110100 \\ \vdots & & \cdots & \cdots & \cdots & \cdots & & \cdots & \cdots & \cdots & \cdots \\ 59 & & a^{59} & a^{51} & a^{43} & a^{35} & & 101111 & 110101 & 111011 & 110100 \\ 60 & & a^{60} & a^{54} & a^{48} & a^{42} & & 100111 & 111010 & 101100 & 010111 \\ 61 & & a^{61} & a^{57} & a^{53} & a^{49} & & 100011 & 011111 & 010101 & 010110 \\ 62 & & a^{62} & a^{60} & a^{58} & a^{56} & & 100001 & 100111 & 111111 & 111110 \\ & & \downarrow & \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow & \downarrow \\ & & S_1 & S_3 & S_5 & S_7 & & S_1 & S_3 & S_5 & S_7 \end{array}$$

We assume that the data elements at the positions $X_1=a^0$, $X_2=a^3$, and $X_3=a^5$ are triggered simultaneously. Then to calculate the Newton power symmetric functions S_1–S_7 it is necessary to perform the mod-2 addition of the elements located at the positions labeled *. This gives

$$\begin{aligned} &S_1=a^0+a^3+a^5=a^{23}, \\ &S_3=a^0+a^9+a^{15}=a^{61}, \qquad (9) \\ &S_5=a^0+a^{15}+a^{25}=a^{35}, \\ &S_7=a^0+a^{21}+a^{35}=a^{61}. \end{aligned}$$

To obtain a high operating speed the syndrome is calculated by using parallel check circuits (Fig. 17). Analysis of the matrix $H_{63,24}$ shows that the number of parity-checking circuits can be decreased considerably if coincident ones are grouped together in the corresponding columns of the check-ratio matrix.[5]

The syndrome code can be generated in only 12 ns if an MC10160 chip with 6 ns delay is used. In our example the number of syndrome bits is $N=18$ for $n=63$ and $t=3$. Therefore, a unitary 63-bit code is transformed into an 18-bit code consisting of three elements of the field $GF(2^6)$, S_1, S_3, and S_5. Here the compression coefficient is $K_c=63/18$. In the examples given below we will use S_7.

A NEW TYPE OF MAJORITY COINCIDENCE SCHEME

Since the syndrome code of a BCH code carries information about the multiplicity t, the next step after data compression is the determination of t. For this we use the following property of the matrix L_t. The $t\times t$- matrix[1,37]

$$L_t = \begin{array}{cccccc} S_1 & 1 & 0 & 0 & 0......0 \\ S_3 & S_1^2 & S_1 & 1 & 0......0 \\ S_5 & S_1^4 & S_3 & S_1^2 & S_1.....0 \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots\dots} \\ S_{2t-1} & S_{2t-2} & S_{2t-3} & S_{2t-4} & S_{2t-5}..S_t \end{array} \tag{10}$$

is nondegenerate if the power symmetric functions depend on t or $t+1$ field elements, and it is degenerate if the S_j depend on fewer than $t-1$ different field elements. In other words, to calculate t it is necessary to calculate the determinant $\det L_t$ in the Galois field. The expressions for the determinants for $t=1$–4 are given in Table VIII.

We see that $\det L_t$ can be zero or any field value. Therefore, the logic expressions for the majority coincidence scheme (MCS) containing n inputs and t outputs have the form[7]

$$\begin{array}{ll} \text{output1} = & \det L_1 \vee \det L_2 \vee \det L_3 \vee \dots \det L_j \vee \dots \det L_t \geqslant 1 \\ \text{output2} = & \det L_2 \vee \det L_3 \vee \dots \det L_j \vee \dots \det L_t \geqslant 2 \\ \text{output3} = & \det L_3 \dots\ \det L_j \vee\ \det L_t \geqslant 3 \\ \dots\dots\dots\dots & \dots\dots\dots\dots\dots\dots\dots\dots \\ \text{output } j = & \det L_j \vee\ \det L_t \geqslant j \\ \dots\dots\dots\dots & \dots\dots\dots\dots\dots\dots\dots\dots \\ \text{output } t = & \det L_t \geqslant t. \end{array} \tag{11}$$

If to the MCS described by Eqs. (11) we add simple schemes for analyzing determinants for 0s and 1s, we can also obtain rigorous equalities, as shown in Fig. 18. It should be noted that such devices can be realized in practice by using tabular methods or the other algorithms discussed above. The circuit shown in Fig. 18 was designed so that it can be realized for large numbers by using PROMs containing $2m$ address inputs. For $t=2$ or 3 and $m=10$–15, programmable logic matrices (PLMs) can be used.

Let us consider an example. Let $t=3$. We have

$$\begin{aligned} &\det L_1 = S_1 = a^{23} \neq 0, \\ &\det L_2 = a^{69} + a^{61} = a^{6} + a^{61} = a^{46} \neq 0, \\ &\det L_3 = a^{138} + a^{69}a^{61} + a^{23}a^{35} + a^{122} \\ &\qquad = a^{12} + a^{4} + a^{58} + a^{59} \neq 0, \end{aligned} \tag{12}$$

but

$$\det L_4 = a^{41} + a^{33} + a^{24} + a^{16} + a^{17} + a^{59} + a^{7} + a^{4} = 0.$$

Thus, the values of t can be calculated rapidly by analyzing the values of the determinants.

DETERMINATION OF EVENT COORDINATES

Using the syndrome properties and algebraic methods of decoding the error positions, while determining the multiplicity t it is possible to simultaneously find the event coordinates X_i $(i=1,2,3,...,t)$. This has two key features.

1) For $t \leqslant 5$ tabular methods can be used.

2) In general, to find the event coordinates it is necessary to solve an equation[1] called the coordinate equation:[6]

$$\begin{aligned} P(X) &= X^t + \sigma_1 X^{t-1} + \sigma_2 X^{t-2} + \dots + \sigma_t \\ &= (X+X_1)(X+X_2)\dots(X+X_j)(X+X_t). \end{aligned} \tag{13}$$

The Newton power symmetric functions S_j and the event coordinates X_i are related as (for the binary case)

$$S_j = \sum_{i=1}^{t} X_i^j, \quad i=1,2,3,...t; \quad j=1,3,5,...(2t-1).$$

Therefore, the Peterson algorithm (applied to the syndrome coding technique) for determining the event coordinates involves three steps.

1) The calculation of the values of S_j.

2) The calculation of the power symmetric functions σ_i. In turn, the values σ_t are related by the Newton S_j expressions (for binary arithmetic):

$$\begin{aligned} &S_1 + \sigma_1 = 0, \\ &S_3 + \sigma_1 S_1^2 + \sigma_2 S_1 + \sigma_3 = 0, \\ &S_5 + \sigma_1 S_1^4 + \sigma_2 S_3 + \sigma_3 S_1^2 + \sigma_4 S_1 + \sigma_5 = 0. \end{aligned}$$

3) Finding the roots X_i of the polynomial $P(X)$. The roots are found by successive substitutions of all possible field elements into Eq. (13). It is easy to see that this method of determining the event coordinates takes a long time. However, for $t \leqslant 5$ there are well known tabular methods of calculating the roots of Eq. (13) (Refs. 38–44). Let us consider the cases $t=2$ and $t=3$ in more detail.

For $t=2$ we have

$$X^2 + \sigma_1 S + \sigma_2 = 0, \tag{14}$$

where $S_1 = \sigma_1$ and $\sigma_2 = (S_1^3 + S_3)/S_1$. Substituting $X = \sigma_1 Y$ into Eq. (14), we obtain

$$Y^2 + Y = \gamma, \tag{15}$$

where $\gamma = \sigma_2/\sigma_1^2$, $X_1 = \sigma_1 Y_1$, $X_2 = \sigma_1 Y_2$, and $Y_2 = Y_1 + 1$.

TABLE VIII.

t	$\det L_t$
1	S_1
2	$S_1^3 + S_3$
3	$S_1^6 + S_1^3 S_3 + S_1 S_5 + S_3^2$
4	$S_1^{10} + S_1^7 S_3 + S_1^5 S_5 + S_1^2 S_3 S_5 + S_1 S_3^3 + S_3 S_7 + S_5^2 + S_1^3 S_7$

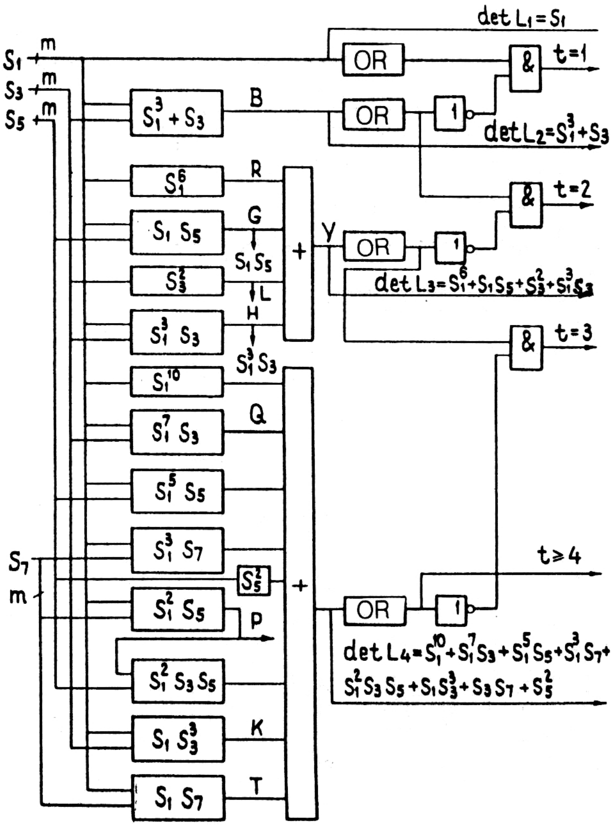

FIG. 18. Circuit for calculating determinants in the field $GF(2^m)$ for $t=1$–4.

The algorithm for solving a quadratic equation (Ref. 14) is well suited for realization by the combined-operation method. It has been shown that if we have the operator $T(r)=0$, then

$$Y_1=\sum_{i=0}^{m-1}\gamma_i y_i. \tag{16}$$

In turn, the values y_i are found from

$$y_i^2+y=\begin{cases}a^i & \mathrm{Tr}(a^i)=0\\ a^i+a^k=1.\end{cases}$$

After simple algebra we obtain the following values of y_0–y_5 for $m=6$ (Ref. 6):

$$y_0=a^0 \quad \text{for } a^i=0,$$

$$y_1=a^{11} \quad \text{for } a^i=a^{36},$$

$$y_2=a^{55} \quad \text{for } a^i=a^{32},$$

$$y_3=a^0 \quad \text{for } a^i=0,$$

$$y_4=a^{33} \quad \text{for } a^i=a^{38},$$

$$y_5=a^{43} \quad \text{for } a^i=a^{19}.$$

In the field $GF(2^6)$,

$$y_1=y_{10}a^0+y_{11}a^1+y_{12}a^2+y_{13}a^3+y_{14}a^4+y_{15}a^5$$

and

$$\gamma=\gamma_0a^0+\gamma_1a^1+\gamma_2a^2+\gamma_3a^3+\gamma_4a^4+\gamma_5a^5. \tag{17}$$

Therefore, from (16) we have

$$y_{10}=\gamma_0,\quad y_{11}=\gamma_0+\gamma_1+\gamma_5,$$

$$y_{12}=\gamma_1+\gamma_2+\gamma_3+\gamma_5,\quad y_{13}=\gamma_0,$$

$$y_{14}=\gamma_0+\gamma_3+\gamma_5,\quad y_{15}=\gamma_0+\gamma_1+\gamma_2+\gamma_4+\gamma_5.$$

In Fig. 19 we show the diagram for solving Eq. (14) (Ref. 43). The speed of operation of the coordinate processor can be estimated from the equation

$$T_{k2}=T_y+2T_s+T_{\mathrm{PROM}\,\gamma},$$

where T_y is the time to multiply two elements in the field $GF(2^m)$, T_s is the time for mod-2 addition, and T_{PROM} is the delay in the PROM. If fast logic circuits are used, it is possible to calculate simultaneously two coordinates in less than 10 ns. Let us consider a specific example. We assume that data elements at the positions $X_1=a^0$ and $X_2=a^2$ are triggered simultaneously. We have $S_1=a^{12}$, $S_3=a^1$, $\sigma_1=a^{12}$, $\sigma_2=a^2$, and $\gamma=a^{41}=101110$. From Eq. (17) we obtain $y_0=y_1=y_3=y_5=1$ and $y_2=y_4=0$. Then Y_1a^{51} and $Y_2=a^{53}$. Finally,

$$X_1=a^{12}a^{51}=a^0 \quad\text{and}\quad X_2=a^{12}a^{53}=a^2.$$

It can be checked that the values of the elements a^0 and a^2 satisfy the equation

$$X^2+a^{12}X+a^2=0.$$

Solution of a cubic equation ($t=3$)

For $t=3$ the syndrome has $3m$ bits. If the value of $3m$ is such that direct decoding is impossible owing to the limited size of the PROM or the PLA, the more economical parallel-sequential tabular method of solving a cubic equation can be used. We have

$$X^3+\sigma_1X^2+\sigma_3=0. \tag{18}$$

After the substitutions $X=\sigma_1+Y$ and $Y=Z(\sigma_1^2+\sigma_2)^{1/2}$, Eq. (18) becomes[40,41]

$$Z^3+Z=C, \tag{19}$$

where

$$C=\frac{\sigma_1\sigma_2+\sigma_3}{(\sigma_1^2+\sigma_2)(\sigma_1^2+\sigma_2)^{1/2}}. \tag{20}$$

To simplify the calculation of the constant C it is useful to express it directly in terms of S_1, S_3, and S_5. We have

$$\frac{\left(\dfrac{S_3^2+S_1^6}{S_1^3+S_3}\right)}{\left(\dfrac{S_1^5+S_5}{S_1^3+S_3}\right)\left(\dfrac{S_1^5+S_5}{S_1^3+S_3}\right)^{1/2}}=D(EE^{1/2})^{-1}.$$

The circuit for the tabular solution of a cubic equation is given in Fig. 20 (Ref. 43). The solution time T_{k3} can be calculated from the relation

$$T_{k3}=5T_{\mathrm{PROM}}+2T_s.$$

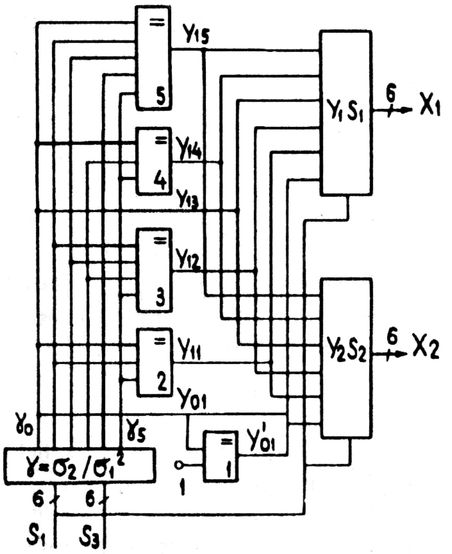

FIG. 19. Circuit for solving a coordinate equation of second degree in $GF(2^m)$.

The authors of Ref. 41 give an algorithm for solving a quartic equation

$$\sigma(X)=X^4+\sigma_1X^3+\sigma_2X^2+\sigma_3X+\sigma_4. \tag{21}$$

After a series of substitutions the problem reduces to the solution of two quadratic equations, which are then solved by the tabular method, as described above. This problem is solved in Ref. 42, but the operations are carried out on logarithms of the field elements. The circuit for a coordinate processor for $t=4$ and $t=5$ is described in Ref. 44. Here for the economical realization of the values of σ_t the author used the Berlekamp transformation method:[45]

$$\sigma_3=R_3+R_1\sigma_2, \tag{22}$$

$$\sigma_4=\frac{R_7+R_5\sigma_2}{R_3}, \tag{23}$$

where

$$\sigma_1=R_1$$

$$\sigma_3=R_3+R_1\sigma_2$$

$$R_5+R_3\sigma_2+R_1\sigma_4=0$$

$$R_7+R_5\sigma_2+R_3\sigma_4=0 \tag{24}$$

$$R_3=S_3+S_1^3=B$$

$$R_5=S_5+S_1^2S_3=V$$

$$R_7=S_7+P+V+S_1^7,$$

with $P=S_1^2S_5$ (Fig. 18). Equation (24) does not contain R_7 and therefore is simpler to calculate than (23):

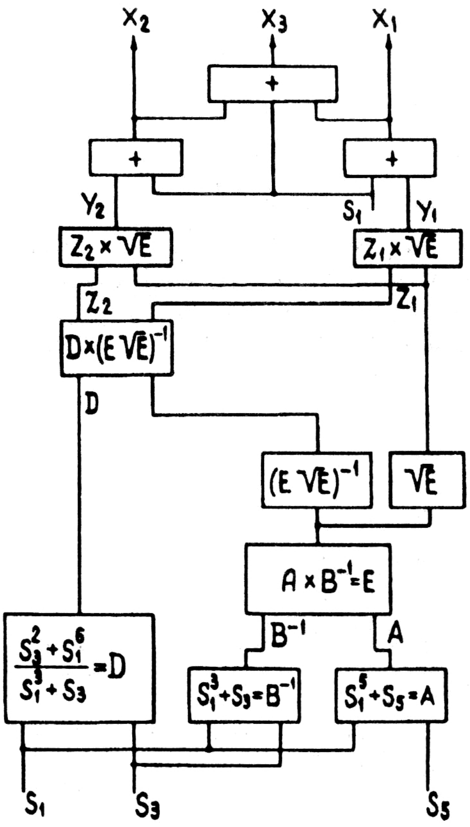

FIG. 20. Circuit for solving a coordinate equation of third degree in $GF(2^m)$.

$$\sigma_4=\frac{R_5+R_3\sigma_2}{R_1}. \tag{25}$$

Therefore, after preliminary calculation of σ_2, which is relatively simple, from (22) and (23) we find σ_3 and σ_4. The expressions for calculating σ_2 for $t=2$–5 are given in Ref. 46. The solution of an equation of fifth degree reduces to the solution of a quadratic and a cubic equation (Refs. 41, 42, and 44).

THE HYBRID METHOD OF SOLVING THE COORDINATE EQUATION

The complexity of the calculations of σ_t and S_j grows rapidly with increasing multiplicity t. The method of solving the general form of the coordinate equation without calculating σ_t (Ref. 47) is therefore of particular interest. This approach makes it possible to design a hybrid processor using the syndrome properties. Depending on the value of t, such a processor can be used to solve the coordinate equation either by the tabular method if $t\leqslant 5$ or by the method of cyclical shifts of the resulting syndrome, where the following properties of the determinants A and d are used. The determinant

$$A=\begin{vmatrix} 1 & 0 & 0......0 \\ S_2 & S_1 & 1......0 \\ S_4 & S_3 & S_2.....0 \\ \multicolumn{3}{c}{\dots\dots\dots\dots\dots\dots\dots\dots} \\ S_{2t-2} & S_{2t-3} & S_{2t-4}...S_{t-1} \end{vmatrix}.$$

is nonzero if the values of S_j, $j=1,2,...,2t$, are formed by power sums of t or $t-1$ different field elements, and it is zero if the values of S_j are formed by power sums of $t-2$ or fewer different field elements. In addition, it is necessary to check the determinant

$$d=\begin{vmatrix} 1 & 1 & 1.......1 \\ S_1 & 1 & 0.......0 \\ S_3 & S_2 & 1.......0 \\ S_5 & S_4 & S_3.....0 \\ \multicolumn{3}{c}{\dots\dots\dots\dots\dots\dots\dots\dots} \\ S_{2t-1} & S_{2t-2} & S_{2t-3}..S_{t-1} \end{vmatrix}.$$

For $t=3$ we have

$$d=\begin{vmatrix} 1 & 1 & 1 & 1 \\ S_1 & 1 & 0 & 0 \\ S_3 & S_2 & S_1 & 1 \\ S_5 & S_4 & S_3 & S_2 \end{vmatrix}$$

$$=S_1^3(1+S_1+S_1^3)+S_3(1+S_1+S_1^2+S_1^3+S_3)$$

$$+S_5(1+S_1)=0.$$

It should be noted that for the characteristic field $2S_2^2=S_1^4$, $S_4=S_1^4$, $S_6=S_3^2$, and so on. For $t=2$ or $t=3$ the determinant

$$A=\begin{vmatrix} 1 & 0 & 0 \\ S_2 & S_1 & 1 \\ S_4 & S_3 & S_2 \end{vmatrix}=S_1^3+S_3$$

is nonzero. However, $A=0$ for $t=1$, since $S_3=S_1^3$. Let us consider an example. To simplify the discussion we assume that $t=3$. Let $X_1=a^{61}$, $X_2=a^{60}$, and $X_3=a^{59}$. The procedure for finding the roots amounts to multiplying S_1, S_3, and S_5 by a^1, a^3, and a^5, respectively, in each cycle. After six such transformations we have

	S_1	S_3	S_5d
initial values	a^{22}	a^1	$a^4\neq 0$
after first cycle	a^{23}	a^4	$a^9\neq 0$
after second cycle	a^{24}	a^7	$a^{14}=0$
after third cycle	a^{25}	a^{10}	$a^{19}=0$
after fourth cycle	a^{26}	a^{13}	$a^{24}=0$
after fifth cycle	a^{27}	a^{16}	$a^{29}\neq 0$
after sixth cycle	a^{28}	a^{19}	$a^{34}\neq 0$.

The circuit for a coordinate processor is shown in Fig. 21. Depending on the value of t, the processor calculates the coordinates in tabular fashion ($t\leqslant 5$) or cyclically. If $t=h$ and $h>5$, the number of cycles will be only $h-5$, after which the processor again returns to the tabular solution for calculating the other five roots. Let us briefly consider the operation of the processor. If the majority coincidence scheme triggers a $t>5$ signal, counter 1 is set up and the generator G begins to generate clock pulses. Counter 2 is used to calculate the value of the degree X_i. In each cycle the value of the syndrome S_j is multiplied by the corresponding constants, and the contents of the multiplication circuits M_1–M_t are added to the register contents. As soon as a 5 appears in counter 1, the comparator is triggered and the process of computing the coordinates by the tabular method begins.

THE USE OF FIRE CODES

Often in sequential readout from coordinate detectors a pulse burst is recorded and it is necessary to determine the coordinate of the center of the cluster (Fig. 22). The signals from the planes are read out using delay lines.[48] An economical scheme for coding the read-out signals using Fire codes was proposed in Ref. 49. Fire codes were specially developed for correcting error bursts in communications and computational technology.[1] Tables of generating polynomials for constructing Fire codes of length $n=15$–1200 are given in Ref. 50. The number of bits in the encoder is equal to the degree r of the generating polynomial $g(X)$. The following generating polynomial is useful for a cluster of length $b=3$ and $n=15$ shifted pulses:

$$g(X)=X^9+X^6+X^5+X^4+X+1.$$

Here the compression coefficient is the ratio n/r. The compression increases with n, since the size of the cluster is limited. In Ref. 51 it was shown that optimal Fire codes exist for $b=3$ and 4. For example, as follows from Ref. 51, for $b=3$ instead of a 9-bit register it is possible to use a 6-bit register and $K_c=4096/14$. As a result, a PROM can be used to decode such a code.

PARALLEL ENCODERS FOR TWO-COORDINATE DETECTORS

By now semiconductor two-coordinate (pixel) detectors have been developed which contain a set of pixels with amplifier-shapers and very simple schemes for determining the limits on the multiplicity using comparators.[52] Priority encoders are used in some applications; their defects have been noted above.[53] In this section we shall show how the syndrome coding technique can be used to encode coordinates and determine the multiplicity of events recorded in two-coordinate detectors. In addition, there is also the possibility of efficiently solving the problem of recognizing "ghosts," since the use of priority encoders even for $t=2$ does not solve the problem when they. are used for separate readout of the X- and Y-coordinate data. If the syndrome coding technique is used for these purposes, two approaches are possible: 1) the use of iterated codes (see below); 2) the use of the algebraic coding theory for BCH codes.[54]

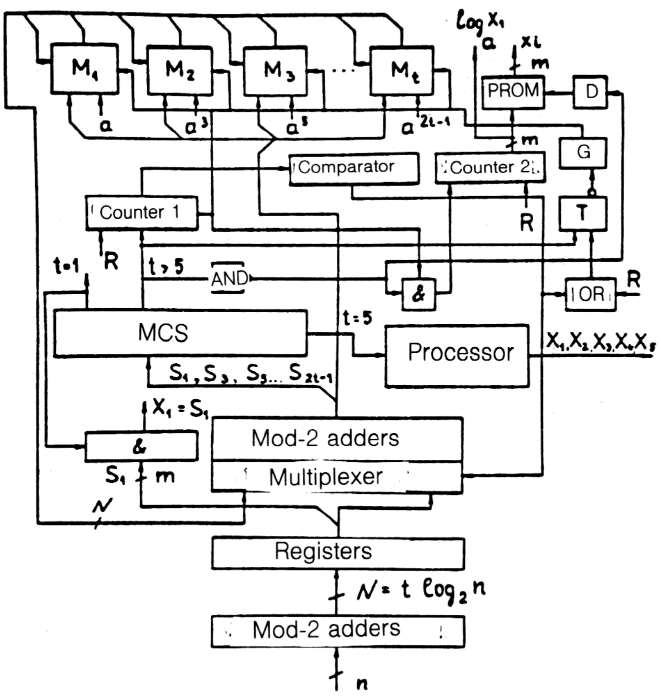

FIG. 21. Circuit for a hybrid coordinate processor: G is a pulse generator, T is a trigger, D is a delay, I is an inverter, & is an AND element, M_1–M_t are devices for multiplication in a Galois field, and R is the setup at 0.

We shall assume that the detector consists of $n=2^m-1=k\times k$ pixels, located in the X, Y plane, and $m>2$. As an example, in Fig. 23 we give the diagram of a detector containing 49 pixels ($k=7$). To simplify matters we assume that $t=2$ and that the pixels with coordinates $X_1=3$, $Y_1=6$ and $X_2=4$, $Y_2=5$ are triggered simultaneously. Otherwise the pixels with coordinates $X_1=3$, $Y_1=5$ and $X_2=4$, $Y_2=6$ are triggered simultaneously. If the data are read out to registers arranged along the X and Y coordinates (not shown in Fig. 23), uncertainties arise in the calculation of the event coordinates, even if priority registers are used. We shall consider two cases for the solution of this problem: 1) the use of separate parallel encoders for each row and each column separately; 2) the use of a single parallel encoder for all n readout channels.

For example, for $t=2$ and $m=3$ we have the following coding matrix $H_{7,2}$:

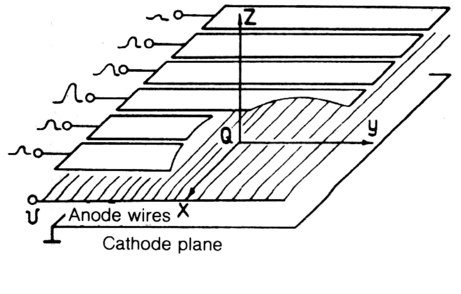

FIG. 22. A two-coordinate detector with cathode readout.

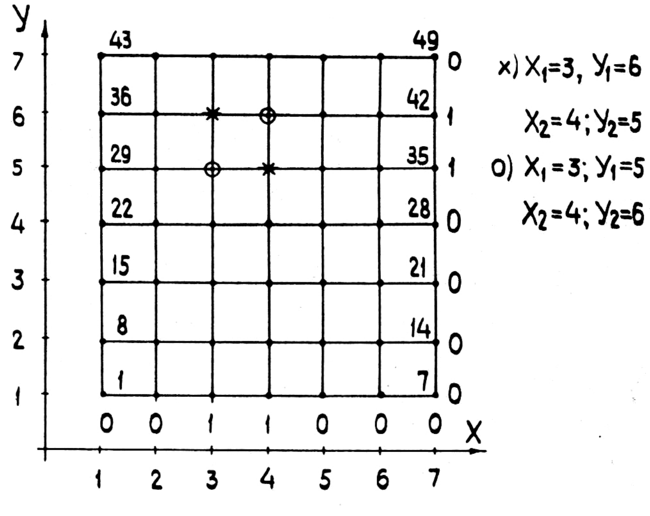

FIG. 23. Schematic depiction of a two-coordinate detector containing 49 pixels; * and O are events.

Number of pixel in row or in column

$$\begin{array}{c} 1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7 \end{array} \quad H_{7,2}=\left|\begin{array}{cc} a^0 & a^0\\ a^1 & a^3\\ a^2 & a^6\\ a^3 & a^2\\ a^4 & a^5\\ a^5 & a^1\\ a^6 & a^4 \end{array}\right| \rightarrow \left|\begin{array}{cc} 100 & 100\\ 010 & 110\\ 001 & 101\\ 110 & 001\\ 011 & 111\\ 111 & 010\\ 101 & 011\\ \downarrow & \downarrow\\ S_1 & S_3 \end{array}\right|$$

In Fig. 24 we give the diagram of a detector in which separate parallel encoders for rows and columns are used. We see from Table IX that in this approach the ghost problem is solved rapidly and simply.

In Fig. 25 we give the circuit for an encoder for $n=7$. In the second coding method all the pixels are numbered in succession and a single parallel encoder is used for all n recording channels. Our example requires an encoder with 49 inputs (Fig. 26), and therefore we choose the following parameters: $m=6$, $n=2^6-1=63$, and $t=2$. Since S_1 and S_3 are elements of the field $GF(2^6)$, they are represented as

$$S_1=S_{10}a^0+S_{11}a^1+S_{12}a^2+S_{13}a^3+S_{14}a^4+S_{15}a^5,$$

$$S_3=S_{30}a^0+S_{31}a^1+S_{32}a^2+S_{33}a^3+S_{34}a^4+S_{35}a^5. \quad (26)$$

Below, we give the coding matrix $H_{49,2}$, which consists of two columns, since $t=2$. For events labeled with the * we have

$$\begin{array}{cc} 101001 & 110011\\ + & +\\ \underline{001101} & \underline{101100}\\ S_1=100100=a^{32} & S_1=011111=a^{57}. \end{array}$$

To estimate the determinant det L_3 we calculate S_5:

$$S_5=(a^{31})^5+(a^{37})^5=a^{2\times 63}a^{29}+a^{2\times 63}a^{59}$$
$$=a^{29}+a^{59}=a^{12}.$$

In Fig. 27 we show the basic circuit for calculating the component S_{10} of the vector S_1. Comparing the two approaches to coding, we see that the first is useful when the number of pixels in the rows (columns) is large.

Signal processing

Let us consider an example to demonstrate the other approach to processing a syndrome code. This is based on the representation of the field elements as polynomials. For our example we have

$$\det L_1=S_1=a^{32}=100100\neq 0,$$

$$\det L_2=(a^{32})^3+a^{57}=a^{63}a^{33}+a^{57}=a^0a^{33}+a^{57}\neq 0,$$

$$\det L_3=a^{192}+a^{96}a^{57}+a^{114}+a^{32}a^{12}$$
$$=a^{189}a^3+a^{63}a^{27}+a^{51}+a^{32}a^{12}$$
$$=a^3+a^{27}+a^{51}+a^{44}=0.$$

It is easy to find, as expected, that det L_3 is equal to zero:

$$\begin{array}{l} 000100 \\ + \\ 011100 \\ + \\ 110101 \\ + \\ \underline{101101} \\ 000000=\det L_3. \end{array}$$

We represent det L_1 and det L_2 in the form

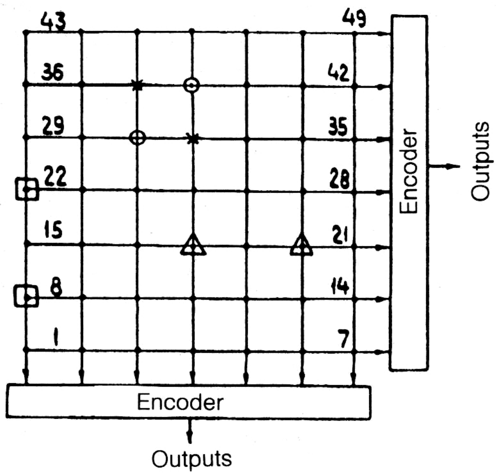

FIG. 24. Schematic depiction of a detector with parallel encoders for each column and each row; *, ○, □, and △ are events.

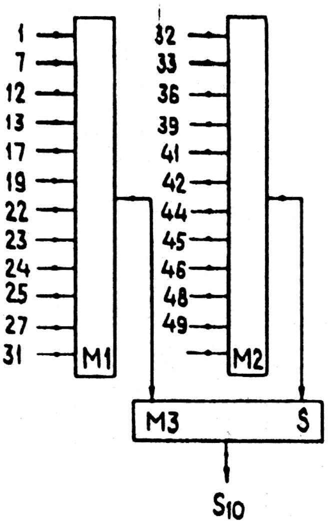

FIG. 25. Parallel encoder for one row (column); $t=2$ and $n=7$.

$$\det L_1=S_1$$
$$=a^0S_{10}+a^1S_{11}+a^2S_{12}+a^3S_{13}+a^4S_{14}+a^5S_{15},$$
$$\det L_2=(S_1^3+S_3)$$
$$=(a^0S_{10}+a^1S_{11}+a^2S_{12}+a^3S_{13}+a^4S_{14}$$
$$+a^5S_{15})^3+(a^0S_{30}+a^1S_{31}+a^2S_{32}+a^3S_{33}$$
$$+a^4S_{34}+a^5S_{35}). \qquad (27)$$

After raising to the power and combining similar terms, we have the following values of the coefficients for the term S_1^3:

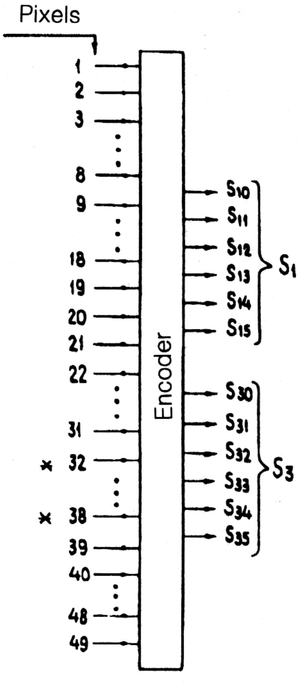

FIG. 26. Circuit for a parallel encoder for $t=2$ and $n=49$.

TABLE IX. Coding of coordinates of triggered pixels.

		Coordinate X							Coordinate Y						
		*	*	o	o	□	□	Δ	*	*	o	o	□	Δ	Δ
S_1	⟶	a^2	a^3	a^2	a^3	a^0	a^0	a^2	a^5	a^4	a^4	a^5	a^0	a^2	a^2
S_3	⟶	a^6	a^2	a^6	a^2	a^0	a^0	a^4	a^1	a^5	a^5	a^1	a^5	a^6	a^6

$$\begin{aligned}
A_{10}&=S_{10}+S_{12}+S_{14}+S_{10}S_{13}+S_{13}S_{15}+S_{11}S_{14}+S_{13}S_{14}\\
&\quad+S_{11}S_{15}+S_{12}S_{15},\\
A_{11}&=S_{12}+S_{10}S_{11}+S_{10}S_{13}+S_{11}S_{13}+S_{12}S_{13}+S_{13}S_{14}\\
&\quad+S_{14}S_{15}+S_{11}S_{14},\\
A_{12}&=S_{14}+S_{10}S_{12}+S_{11}S_{13}+S_{12}S_{14}+S_{12}S_{15}+S_{14}S_{15}\\
&\quad+S_{10}S_{11}+S_{10}S_{14}+S_{11}S_{15},\\
A_{13}&=S_{11}+S_{13}+S_{15}+S_{11}S_{14}+S_{12}S_{13}+S_{12}S_{14}+S_{12}S_{15}\\
&\quad+S_{13}S_{15}+S_{14}S_{15}+S_{10}S_{13}+S_{10}S_{14}, \qquad (28)\\
A_{14}&=S_{13}+S_{14}S_{15}+S_{10}S_{12}+S_{10}S_{14}+S_{10}S_{15}+S_{11}S_{12}\\
&\quad+S_{11}S_{14}+S_{12}S_{14}+S_{12}S_{15}+S_{13}S_{14},\\
A_{15}&=S_{15}+S_{11}S_{12}+S_{11}S_{13}+S_{11}S_{15}+S_{12}S_{13}+S_{13}S_{15}.
\end{aligned}$$

Finally, we obtain the following Boolean expressions for det L_2:

$$\begin{aligned}
\det L_2&=\det L_{20}+\det L_{21}+\det L_{22}+\det L_{23}+\det L_{24}\\
&\quad+\det L_{25}.\\
\det L_{20}&=A_{10}+S_{30}\\
\det L_{21}&=A_{11}+S_{31}\\
\det L_{22}&=A_{12}+S_{32} \qquad (29)\\
\det L_{23}&=A_{13}+S_{33}\\
\det L_{24}&=A_{14}+S_{34}\\
\det L_{25}&=A_{15}+S_{35}.
\end{aligned}$$

TABLE X. Coding matrix $H_{50,2}$.

Pixel number	$GF(2^6)$-elements	Cubes of these elements	Pixel number	$GF(2^6)$-elements	Cubes of these elements
1	100000	100000	26	010001	101000
2	010000	000100	27	101000	000101
3	001000	110000	28	011100	111100
4	000100	000110	29	001110	110111
5	000010	101000	30	000111	100010
6	000001	000101	31	110011	011100
7	110000	111100	* 32	101001	110011
8	011000	110111	33	100100	010010
9	001100	100010	34	010010	011010
10	000110	011100	35	001001	011011
11	000011	110011	36	110100	010111
12	110001	010110	37	011010	100110
13	101000	011010	* 38	001101	101100
14	010100	011011	39	110110	110101
15	001010	010111	40	011011	111010
16	000101	100110	41	111101	011111
17	110010	101100	42	101110	100111
18	011001	110101	43	010111	100000
19	111100	111010	44	111011	000100
20	011110	011111	45	101101	110000
21	001111	100111	46	100110	000011
22	110111	100000	47	010011	101000
23	101011	000100	48	111001	000101
24	100101	110000	49	101100	111100
25	100010	000011			

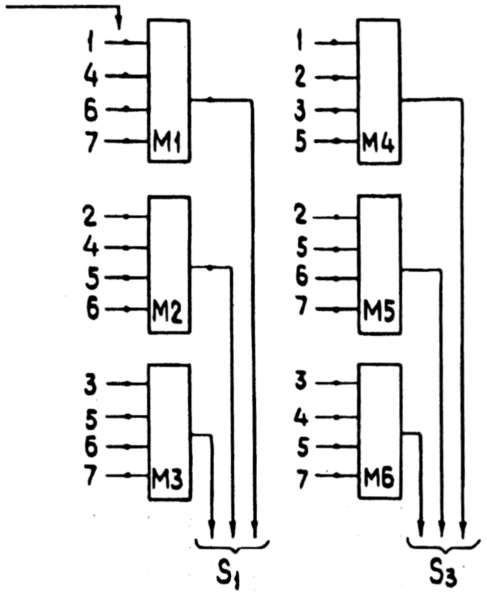

FIG. 27. Basic circuit for calculating the component S_{10} of the vector S_1.

In Fig. 28 we show the basic circuit of a processor for determining the coordinates X_1 and X_2 and multiplicity $t \leqslant 2$. A PROM or a PLA can be used to transform the field elements S_1 and S_3 into natural binary code. For obtaining the maximum operating speed, it is sufficient to use coincidence matrices and parity-checking schemes in the construction of the MCS. Such devices can be realized as chips with large-scale integration. All the logic couplings in the processor are described by means of logic equations, so that the design process can be automated for large n and t.

The rate of operation of the MCS, T_M, can be calculated from the expression

$$T_M = 2T_p + 3T_{AND},$$

where T_p is the delay in the parity-checking circuit and T_{AND} is the delay in the AND logic element. If fast circuits are used, for sufficiently large n and t the value of T_M will be less than 10 ns. Therefore, to construct an MCS using the syndrome coding technique it is necessary to perform the following procedures.

1. In accordance with the number of inputs n, an irreducible polynomial of degree m is selected (Ref. 1) and the $2^m - 1$ nonzero elements of the field $GF(2^m)$ are calculated. For $m = 4$–10 the following polynomials can be recommended:

$$X^4+X+1,\ X^5+X^2+X+1,\ X^6+X+1,\ X^7+X^3+1,$$
$$X^8+X^4+X^3+X^2+1,\ X^9+X^4+1,\ \text{and}\ X^{10}+X^3+1.$$

2. The matrix of check ratios $H_{n,t}$ is constructed for a given $t \leqslant n/2$. The number of columns in the matrix must be t.

3. In the matrix $H_{n,t}$ the field elements are replaced by their binary equivalents.

4. The order-t determinants are calculated.

5. If PROMs or PLMs are not used to calculate the determinant, all the values of the field elements are repre-

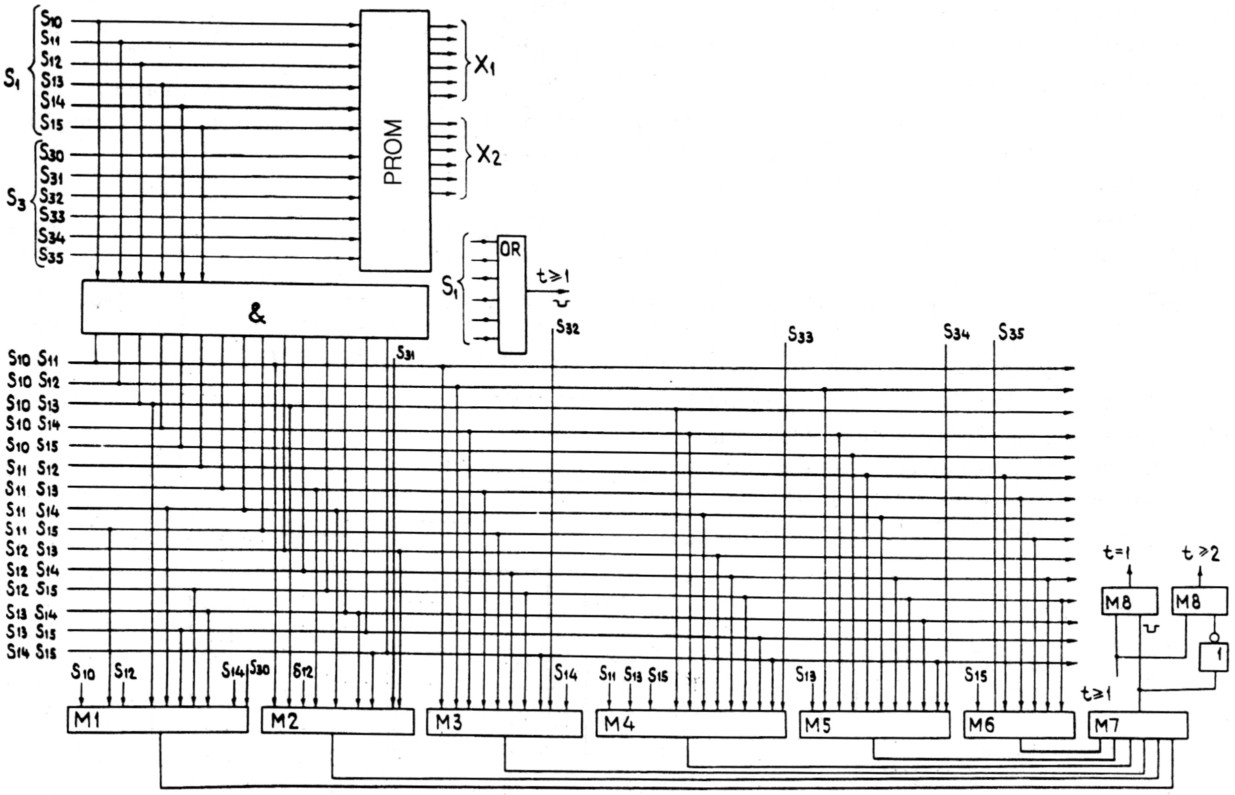

FIG. 28. Basic majority coincidence circuit for $T \leqslant 2$; M1–M7 are MC10160 chips, and M8 is an MC10102 chip.

sented as polynomials of degree $m-1$ and the required calculations are carried out in order to obtain the Boolean expressions in the AND and exclusive-OR basis.

6. The basic MCS circuit is designed in accordance with the expressions (29). Here it is best to use semiconductor technology. A hybrid processor can be used to calculate the event coordinates.

Let us continue the discussion of our example. For $t=1$, $S_1=a^{31}=101001$, i.e., $S_{10}=S_{12}+S_{15}=1$ and $S_{11}=S_{13}=S_{14}\neq 0$. Then $S_3=a^{30}=110011$, i.e., $S_{30}=S_{31}=S_{34}=S_{35}=1$ and $S_{32}=S_{33}=0$. It can be checked that det $L_1=S_1\neq 0$, but det $L_2=0$. In this case the AND logic element (Fig. 28) is open, and signals $t>1$ and $t=1$ are shaped at the MCS output. It can be checked that signals $t\geqslant 1$ and $t\leqslant 2$ are formed for $t=2$.

USE OF THE THEORY OF THE REED–SOLOMON CODE

In some experiments it is necessary not only to record the coordinates of individual triggered position-sensitive detectors, but also to determine the number of clusters, their coordinates, and even their shapes as rapidly as possible. For example, from the size of a cluster recorded in a calorimeter it is possible to determine the energy of the interacting particles. In coordinate detectors, recording the cluster center makes it possible to increase the accuracy of determining the event coordinates. In such cases it is possible to use the theory of error-burst correcting codes. It is well known that in coding theory an error burst of length β is defined as an error vector in which all ones are enclosed in a sequence of symbols with the condition that the end symbols of this sequence are ones. For example, bursts of length 6 might look as follows:

$$\underbrace{0011111110000}_{\text{Burst 1}}\ \underbrace{001000010000}_{\text{Burst 2}}\ \underbrace{001100110000}_{\text{Burst 3}},\ \text{etc.}$$

In Ref. 55 the author suggested that such events be recorded using the theory of Reed–Solomon (RS) error-burst correcting codes.[56] If it is assumed that the cluster length is $\beta\leqslant m$, where m is the degree of the irreducible polynomial, then the compression coefficient is $(2^m-1)/2t$, where 2^m-1 is the number of information symbols and $2t$ is the number of check symbols of the RS code, since they have a length of $n=2^m-1$ symbols, each symbol containing m bits. Among these symbols there are 2^m-1-2t information ones and $2t$ check ones, where t is the number of erroneous symbols corrected by the code. The general form of the check matrix for the RS code, which in our case is simultaneously the coding matrix, is

$$H=\left|\begin{array}{cccc} a^0 & a^0 & a^0.. & ....a^0 \\ a^1 & (a^1)^2 & (a^1)^3.. & ...(a^1)^t \\ a^2 & (a^2)^2 & (a^2)^3.. & ...(a^2)^t \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ a^{n-1} & (a^{n-1})^2 & (a^{n-1})^3 & (a^{n-1})^t \end{array}\right|. \qquad (30)$$

For $t=2$ and $m=4$ we have

$$\begin{array}{c} \text{Groups} \\ \text{of inputs} \\ \\ {*0} \\ 1 \\ {*2} \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{array} \quad H_{60,16}=\left|\begin{array}{cccc} a^0 & a^0 & a^0 & a^0 \\ a^1 & a^2 & a^3 & a^4 \\ a^2 & a^4 & a^6 & a^8 \\ a^3 & a^6 & a^9 & a^{12} \\ a^4 & a^8 & a^{12} & a^1 \\ a^5 & a^{10} & a^0 & a^5 \\ a^6 & a^{12} & a^3 & a^9 \\ a^7 & a^{14} & a^6 & a^{13} \\ a^8 & a^1 & a^9 & a^2 \\ a^9 & a^3 & a^{12} & a^6 \\ a^{10} & a^5 & a^0 & a^{10} \\ a^{11} & a^7 & a^3 & a^{14} \\ a^{12} & a^9 & a^6 & a^3 \\ a^{13} & a^{11} & a^9 & a^7 \\ a^{14} & a^{13} & a^{12} & a^{11} \end{array}\right|. \qquad (31)$$

In the matrix (31) the quantities a^0–a^{14} are elements of the Galois field $GF(2^4)$.

For definiteness we assume that the multichannel detector has 60 measurement channels, split into 15 groups of 4 channels. In turn, the groups of measurement channels are numbered by the degrees of the elements of the Galois field $GF(2^4)$, and the number of the groups of channels in which a cluster appeared is labeled *. As in algebraic coding theory, we assume that the nonzero component of the coordinate vector of the triggered position-sensitive detectors $e(X)$ is specified by a pair of elements Y_i, the cluster shape symbol, and X_i, the cluster coordinate. If there were t events, the vector $e(X)$ has t nonzero components, so that to describe such events it is necessary to have t pairs X_i, Y_i. Then we have

$$S_j=\sum_{i=1}^{t} Y_iX_i^j,\quad 1\leqslant j\leqslant 2t. \qquad (32)$$

The methods described above can be used to solve Eq. (32). The theorem given in Ref. 1 can be used to determine the number of clusters recorded. The matrix

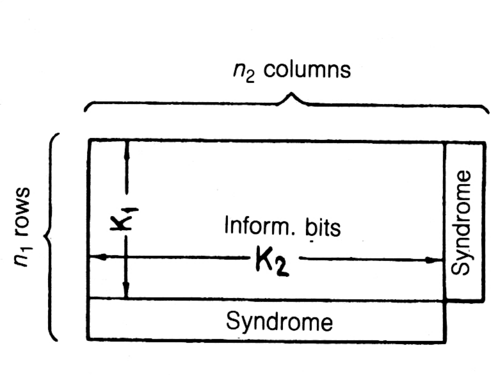

FIG. 29. Structure of a two-dimensional iterated code.

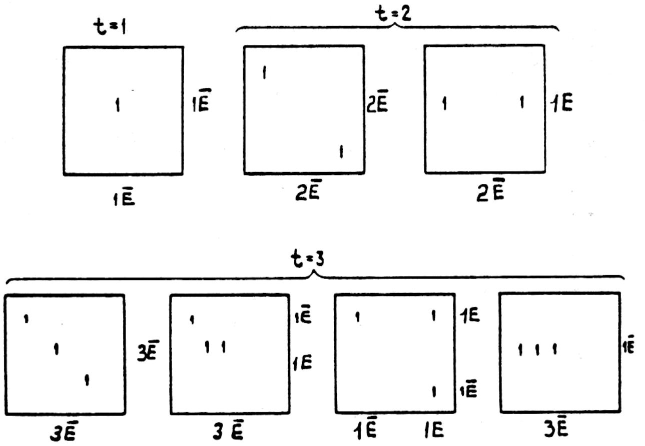

FIG. 30. Form of events in the two-dimensional plane; $t=1$–3; E is an "even" and $\bar{E}$ is an "odd."

$$L_t=\begin{vmatrix} S_1 & S_2 & S_3\ldots\ldots S_t \\ S_2 & S_3 & S_4\ldots\ldots S_{t+1} \\ \cdots & \cdots & \cdots \\ S_t & S_{t+1} & S_{t+2}\ldots S_{2t+1} \end{vmatrix}$$

is nondegenerate if the S_j are formed exactly from t different nonzero pairs (X_i,Y_i), and the matrix (L_t) is degenerate if the S_j are formed from fewer than t nonzero pairs (X_i,Y_i). In other words, the properties of the determinant det L_t are such that if, for example, $t=1$, then det $L_1\neq 0$, but in return all the other determinants of higher order are zero. If $t=2$, then det $L_1\neq 0$, det $L_2\neq 0$, but det $L_3=0$, and so on. For $t=3$ we have

$$L_3=\left|\begin{array}{ccc} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{array}\right| \begin{array}{l} \det L_1=S_1 \\ \det L_2=S_1S_3+S_2^2 \\ \det L_3=S_1S_3S_5+S_1S_4^2+S_2^2S_5+S_3^3 \end{array}. \tag{33}$$

Let us go on with our example. We assume that $t=2$ clusters have been recorded in the multichannel detector at the positions $X_1=a^0$ and $X_2=a^2$, with $Y_1=a^7=1101$ and $Y_2=a^{11}=0111$. Let us consider two cases.

a) The cluster $a^{11}=0111$ was located at $X_1=a^2$. Then from (32) we have

$$S_1=a^{11}a^2=a^{13},\quad S_2=a^{11}a^4=a^{15}=a^0,$$

$$S_3=a^{11}a^6=a^2,\quad S_4=a^{11}a^8=a^4.$$

In addition, from Eq. (33) we have

$$\det L_1=a^{13}\neq 0,\quad \det L_2=a^{13}a^2+a^0=0,$$

$$\det L_3=a^{13}a^2a^6+a^{13}a^8+a^0a^6+a^6=0.$$

b) We assume that $X_1=a^0$, $Y_1=a^7$ and $X_2=a^2$, $Y_2=a^{11}$, i.e., $t=2$. Furthermore, we have

$$S_1=a^7a^0+a^{11}a^4=a^5,\quad S_2=a^7a^0+a^{11}a^4=a^9,$$

$$S_3=a^7a^0+a^{11}a^6=a^{12},\quad \text{and}\quad S_4=a^7a^0+a^{11}a^8=a^3.$$

To find the values of X_1 and X_2 it is necessary to solve the quadratic equation

$$X^2+\sigma_1X+\sigma_2=0. \tag{34}$$

For the special case $t=2$ we have

$$\sigma_1=\frac{S_3S_2^2+S_1S_2S_4}{S_2^3+S_1S_2S_3},\quad \sigma_2=\frac{S_2S_4+S_3^2}{S_2^2+S_1S_3}, \tag{35}$$

and from Eq. (32) we obtain a relation between the S_j, X_i, and Y_i:

$$S_1=X_1Y_1+X_2Y_2\quad \text{and}\quad S_3=X_1^3Y_1+X_2^3Y_2. \tag{36}$$

From Eq. (35) we have $\sigma_1=a^8$ and $\sigma_2=a^2$. It can be checked that for these values of σ_1, σ_2 and $X_1=a^0$, $X_2=a^2$ Eq. (34) becomes an identity. Similarly, Eqs. (36) become an identity for the given values of S_1, S_3 and X_1, X_3. It is therefore possible to calculate rapidly the multiplicity t, the cluster coordinates X_i, and the cluster shapes Y_i by using the syndrome code S_j and a special-purpose processor.[55]

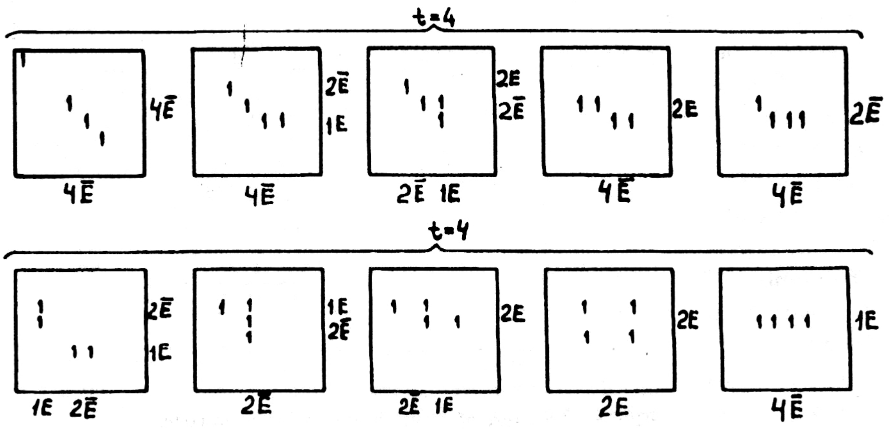

FIG. 31. Value of the parities for $t=4$.

THE USE OF ITERATED CODES

In some cases it makes sense to use so-called iterated codes[57,58] instead of codes with algebraic decoding methods. The former are convenient for coding information recorded in two-coordinate detectors. Any known correcting codes can be used as the initial codes for constructing iterated codes. We shall discuss the method of constructing a two-dimensional iterated code for the following example. First, the information bits $X_{ij}=0$, 1 are represented as a matrix:

$$\begin{array}{r|cccc|}
\multicolumn{1}{r}{} & & & \text{Information symbols } \downarrow & \multicolumn{1}{c}{\text{Checks over rows } \downarrow} \\
 & X_{11} & X_{12} & X_{13}\dots\dots X_{ij}\dots X_{1m} & \sum_{j=1}^{m} X_{1j} \\
 & X_{21} & X_{22} & X_{23}\dots\dots X_{2j}\dots X_{2m} & \sum_{j=1}^{m} X_{2j} \\
 & \dots\dots & \dots\dots & \dots\dots\dots\dots\dots\dots & \dots\dots \\
 & X_{\alpha 1} & X_{\alpha 2} & X_{\alpha 3}\dots\dots X_{\alpha j}\dots X_{\alpha m} & \sum_{j=1}^{m} X_{am} \\
\text{Checks over columns} & \sum_{i=1}^{\alpha} X_{i1} & \sum_{i=1}^{\alpha} X_{i2} & \sum_{i=}^{\alpha} X_{i3}\dots \sum_{i=1}^{\alpha} X_{ij}\dots \sum_{i=1}^{\alpha} X_{im} &
\end{array} . \tag{37}$$

After this, checks in rows and columns are added. In Fig. 29 we show the diagram of an iterated code. An important feature is the fact that the total code distance d of a γ-dimensional iterated code is $d=d_1d_2...d_\gamma$, where γ is the number of codes used for the iteration. Recalling the familiar relation $t=(d-1)/2$, for large numbers n this code can be used for the economical recognition of complicated events with high multiplicity. The simplest iterated code is a code of the OR–OR and "even–even" type ($d=2\times2=4$). Analysis has shown[57] that an OR–OR code is useful in scintillation hodoscopes for economizing on PMs in recording single-particle events with clusters. The check ratios of even–even codes are calculated using mod-2 adders, so that PMs are not suitable for this, since these devices act as signal amplifier-mixers. However, an iterated even–even code can be used effectively if the coded signals have logic levels. Let us consider two figures. In Fig. 30 we show pictures of events in the two-plane for $t=1$–3. The event positions are absolute. We shall count the number of events by computing the number of "even" or "odd" signs separately in the X and Y coordinates. As can be seen from Fig. 31, for $t=1$ we obtain a single "odd" in the X coordinate and a single "odd" in the Y coordinate. For $t=2$, 3, and 4 there are 2, 4, and 10 different pictures, respectively. In Fig. 31 we show the analogous pictures for $t=4$ (E is "even" and $\bar{E}$ is "odd"). There is no need to give the symmetric pictures, which lead to the same result. Therefore, if parallel counters are used to count the number of "even" and "odd" signals, it is possible to construct an economical and fast device[59] for recording the multiplicity in one- and two-coordinate detectors. For example, let us assume that a two-coordinate detector contains $31\times31=961$ pixels (31 rows and 31 columns). To make things simpler, in Fig. 32 we give the main part of the scheme for a single row (column). Since zero is an even number, a certain number of OR logic elements is required in order not to read out zero. The operating speed of such a counter T_c can be calculated from the expression

$$T_{\rm c}=T_{\rm p}+2T_{\rm AND}+T_{\rm ctr}+T_{\rm d},$$

where $T_{\rm p}$ is the delay in the parity-checking scheme, $T_{\rm AND}$ is the delay in the AND logic element, $T_{\rm ctr}$ is the operating speed of the (31,5) counter, and $T_{\rm d}$ is the delay in the decoder. If ECL logic is used, T_c is less than 43 ns! (Ref. 60). Naturally, to record large values of t it is necessary to choose more powerful codes (codes with large d). There are two approaches to choosing codes with large d. 1) Two more diagonal checks are added to the simplest two-

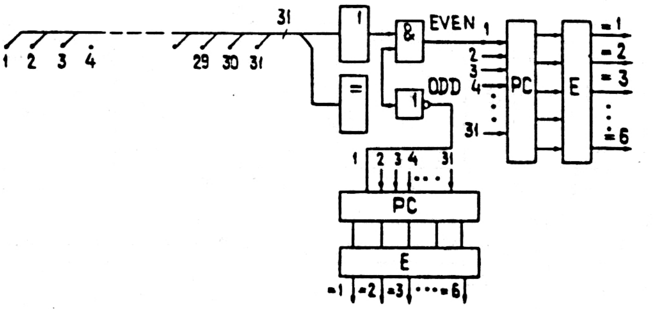

FIG. 32. Part of the circuit for a parallel counter for a two-coordinate detector containing 961 pixels: PC is a parallel counter, 1 is an OR element, & is an AND element, and E is a decoder.

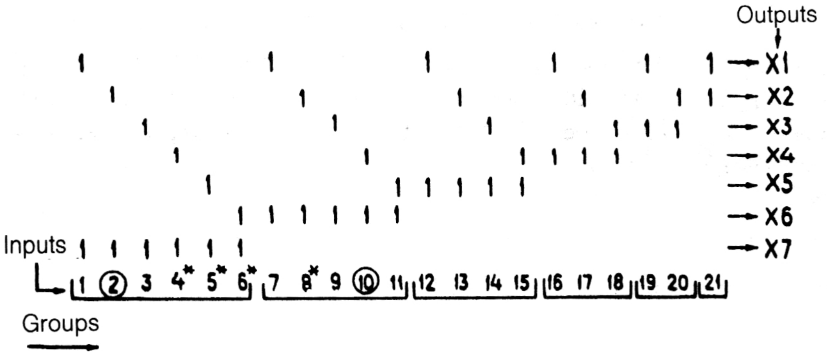

FIG. 33. Coding matrix $H_{28,7}$.

dimensional iterated code. 2) Codes with large code distances are used. It is known that the modified Hamming code has a code distance $d=4$. Then the two-dimensional Hamming–Hamming iterated code has $d=16$ and $t=15$. In addition, it follows from the theory that such a code can be used to record simultaneously $\nu=t/2$ coordinates.

Let us summarize. If we compare the efficiency of codes with an algebraic structure, such as BCH codes and iterated codes, we see the following. Codes with an algebraic structure are relatively easy to decode. Moreover, since the theory of the finite Galois field provides the theoretical basis for such codes, in addition to performing encoding and decoding processes on the data it is easy to realize various types of logic and algebraic operations. The syndrome of an iterated code is a random set of binary symbols, and therefore tabular methods based on PROMS and PLMs are best for decoding it.

THE USE OF SUPERPOSITION CODES

In the examples that we have considered the syndrome was usually calculated using mod-2 adders, the construction of which requires inverters, and the input and output signals must be logic signals. There is another class which is interesting from both the theoretical and, especially, the practical points of view, namely, that of superposition codes.[61] The distinctive feature of such codes is the fact that ordinary signal amplifier-mixers like PMs can be used for the syndrome-calculation circuits. In other words, the coded signals can be both electrical signals of arbitrary shape and amplitude and light signals. In particular, the author has shown that such popular codes as the ordinary Hamming code and the Gray code are superposition codes.[64]

Below, we will show how a series of such codes and encoding devices developed by the author can be used to construct efficient event-selection circuits. The operation of one of these devices is based on a property of the coding matrix C_N^2 (the number of combinations of N elements two at a time; Refs. 62 and 63). The matrix contains $n=C_N^2$ columns and N rows. Each column contains only two ones (the signal branching coefficient is equal to two), and the compression coefficient is $K_c=C_N^2/N$. For example, let $n=28$; then $N=7$. In Fig. 33 we show the coding matrix $H_{28,7}$, and in Fig. 34 we show how bits 1, 3, 4, and 8 of the syndrome are calculated. Flexible light guides can be used as signal splitters. By means of this coding scheme it is possible, with the optimal value of the compression coefficient, to record uniquely single-particle events with three clusters, as is easily checked by computing the Boolean sums of two and three adjacent columns.[64] In addition, such a coding scheme can be realized as a mask, as shown in Fig. 35. Let us consider two cases.[64] 1) A single particle without a cluster is recorded in the hodoscopic plane, i.e., $t=1$. Then the weight of the syndrome is two. Since all the columns of the coding matrix are different, the 7-bit syndrome code carries information both about the fact that $t=1$, and about the coordinate of the triggered position-sensitive detector, which is simple to decode by using a PROM. 2) If a double or triple cluster is recorded, the syndrome weight is three or four, and the values of the codes differ for all combinations of one, two, and three triggered adjacent data elements.

A cluster counter

The coding matrix $H_{64,8}$ consisting of eight repeating unit matrices of eighth order I_8 can be used to construct an economical counter for clusters with $1\leqslant b\leqslant 8$. In Fig. 36 we show the circuit for the counter. The PROM module is programmed so that if the syndrome weight is $w=1$, then $b=1$, while if the weight is $w=2$, then $b=2$, and so on. In general, to construct a counter with $b=z$ it is necessary to use a unit matrix of order z.

Superposition iterated codes

To increase the code distance of superposition codes, for the iteration it is possible to use the popular Hamming

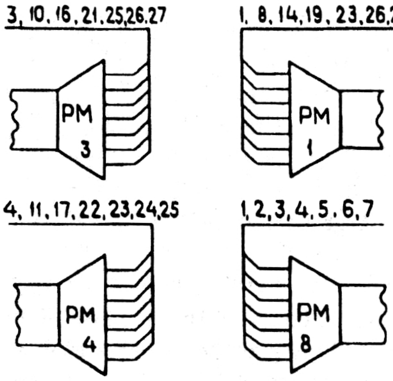

FIG. 34. Circuit for calculating bits 1, 3, 4, and 8 of a syndrome; PM is a photomultiplier.

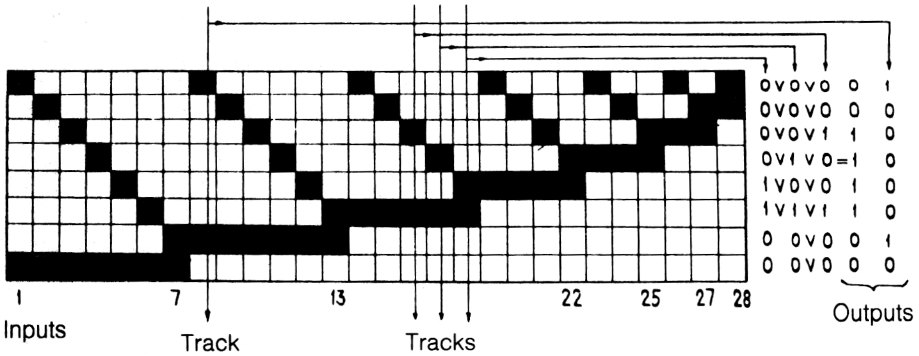

FIG. 35. Detector mask for $t=1$; $n=28$ and $N=7$.

code ($d=3$) and the Gray code ($3\leqslant d\leqslant 4$) in conjunction with a simple coding scheme, when the checks for one of the coordinates are calculated using OR logic elements.[57] Let us assume that a two-coordinate detector contains 225 pixels arranged as a square matrix containing $k=15$ rows and 15 columns. For example, for the rows we calculate the check symbols in accordance with the (15,4) Gray code, and for the columns we use ordinary OR elements ($d=2$). As a result, we obtain the following coding matrix for a single row, the columns of which are essentially 4-bit words $H_{15,4}$ of Gray code:

$$\begin{array}{rc|l} & & N\\ & & \downarrow\\ & \left|\begin{matrix}110011001100110\\011110000111100\\000111111110000\\000000011111111\end{matrix}\right| & \begin{matrix}1\\2\\3\\4\end{matrix}\\ H_{15,4}= & & \\ & n\ 1 \ldots\ldots\ldots\ldots 15 & \end{array}$$

In Fig. 37 we give the basic circuit for calculating the syndrome for a single line. On the whole, this scheme requires 15 OR elements for 15 inputs and 4×15 OR elements for 8 inputs. The code distance of such an iterated superposition code is 8. It should be noted that in practice it is useful to first model events for each particular experiment and detector, and then the corresponding code for fast and optimal coding can be worked out.

CONCLUSIONS

Current and future experiments in high- and superhigh-energy physics are tending to be performed under conditions which are contradictory from the viewpoint of electronics and computational technology: it is necessary to select useful events at a very high rate while recording an enormous amount of information and seeking rare events at the level of a large background. It has become obvious that the solution of this complicated problem requires a nontraditional approach to the design of fast, multilevel systems. The syndrome coding technique essentially amounts to the following. Many problems in the selection of useful events which have traditionally been solved using ordinary binary arithmetic, in which the series of operations is rather complex, can be solved using the algebra of finite fields. Here, before analyzing events, the data read out from multichannel charged-particle detectors is first compressed in the ratio $n/\log_2 n$.

In addition, it has been shown that by using the techniques of calculation in a Galois field $GF(2^m)$ and performing analytic calculations by computer, it is possible to synthesize rather complicated devices such as program-controlled logic modules, parallel encoders, majority coin-

$$\begin{array}{r|c|c|c|c|c|c} & & & & & & N\\ & & & & & & \downarrow\\ & 10000000 & & & & & 1\\ & 01000000 & & & & & 2\\ & 00100000 & & & & & 3\\ & 00010000 & & & & & 4\\ H_{64,8} = & 00000100 & I_8 & \ldots\ldots\ldots & I_8 & & 5\\ & 00000010 & & & & & 6\\ & 00000010 & & & & & 7\\ & 00000001 & & & & & 8\\ n \longrightarrow & 1 & \multicolumn{4}{l}{\ldots\ldots\ldots\ldots\ldots\ldots 57\ldots\ldots 64} & \end{array}$$

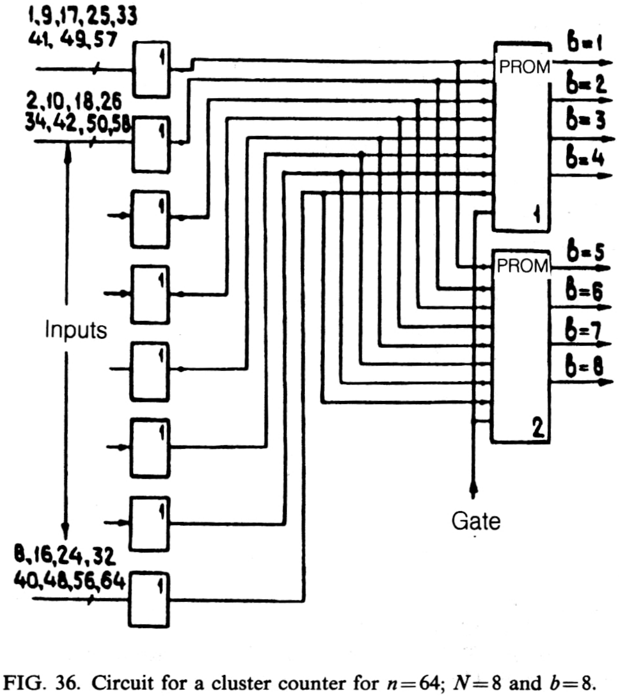

FIG. 36. Circuit for a cluster counter for $n=64$; $N=8$ and $b=8$.

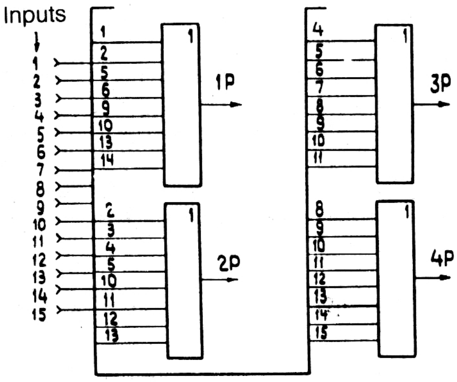

FIG. 37. Circuit for calculating a syndrome for a single row: 1 is an OR element, and 1P–4P are outputs.

cidence schemes, and special-purpose processors for fast selection of physical events. It is also important that the realization of a universal dynamically programmable module will make it possible in future to design complicated trigger systems which do not contain mechanical contacts and can rapidly be reprogrammed for solving different physical problems.

APPENDIX

ELEMENTS OF THE GALOIS FIELD MODULO X^6+X+1

$a^0=100000$	$a^{21}=110111$	$a^{42}=010111$
$a^1=010000$	$a^{22}=101011$	$a^{43}=111011$
$a^2=001000$	$a^{23}=100101$	$a^{44}=101101$
$a^3=000100$	$a^{24}=100010$	$a^{45}=100110$
$a^4=000010$	$a^{25}=010001$	$a^{46}=010011$
$a^5=000001$	$a^{26}=111000$	$a^{47}=111001$
$a^6=110000$	$a^{27}=011100$	$a^{48}=101100$
$a^7=011000$	$a^{28}=001110$	$a^{49}=010110$
$a^8=001100$	$a^{29}=000111$	$a^{50}=001011$
$a^9=000110$	$a^{30}=110011$	$a^{51}=110101$
$a^{10}=000011$	$a^{31}=101001$	$a^{52}=101010$
$a^{11}=110001$	$a^{32}=100100$	$a^{53}=010101$
$a^{12}=101000$	$a^{33}=010010$	$a^{54}=111010$
$a^{13}=010100$	$a^{34}=001001$	$a^{55}=011101$
$a^{14}=001010$	$a^{35}=110100$	$a^{56}=111110$
$a^{15}=000101$	$a^{36}=011010$	$a^{57}=011111$
$a^{16}=110010$	$a^{37}=001101$	$a^{58}=111111$
$a^{17}=011001$	$a^{38}=110110$	$a^{59}=101111$
$a^{18}=111100$	$a^{39}=011011$	$a^{60}=100111$
$a^{19}=011110$	$a^{40}=111101$	$a^{61}=100011$
$a^{20}=001111$	$a^{41}=101110$	$a^{62}=100001$
		$a^{63}=a^0=100000$

[1] W. W. Peterson, *Error-Correcting Codes* (MIT Press, Cambridge, Mass., 1961) [Russian transl., Mir, Moscow, 1964].

[2] T. C. Ancheta, IEEE Trans. Inf. Theory **IT-22**, 432 (1976).

[3] R. A. Frohwerk, Hewlett Packard J. **28**, No. 9, 2 (1977).

[4] N. M. Nikityuk, R. S. Radzhabov, and M. D. Shafranov, Nucl. Instrum. Methods **155**, 485 (1977).

[5] N. M. Nikityuk, R. S. Radzhabov, and M. D. Shafranov, Prib. Tekh. Eksp. No. 4, 95 (1978) [Instrum. Exp. Tech.].

[6] N. M. Nikityuk, Preprint R11-84-484, JINR, Dubna (1984) [in Russian].

[7] Z. I. Gaĭdamaka, V. A. Kalinnikov, N. M. Nikityuk *et al.*, Preprint R13-82-628, JINR, Dubna (1982) [in Russian].

[8] N. M. Nikityuk, Preprint R10-87-254, JINR, Dubna (1987) [in Russian].

[9] N. M. Nikityuk, Preprint E10-90-184, JINR, Dubna (1990); in *Proc. of the AAECC-8 Conf.*, Tokyo, 1990, Lecture Notes in Computer Science [Springer-Verlag, 1990], Vol. 508, p. 144.

[10] L. Gustafson and E. A. Hagberg, Nucl. Instrum. Methods A **265**, 521 (1988).

[11] I. S. Reed and T. K. Truong, IEEE Trans. Inf. Theory **IT-21**, 657 (1977).

[12] T. L. Both, IRE Trans. Circuit Theory **CT-10**, 279 (1975).

[13] K. S. Menger, IEEE Trans. Comput. **C-18**, 241 (1969).

[14] E. R. Berlekamp, *Algebraic Coding Theory* (McGraw-Hill, New York, 1968) [Russian transl., Mir, Moscow, 1970].

[15] R. Y. Kain, IEEE Trans. Comput. **C-19**, 249 (1970).

[16] W. R. English, IEEE Trans. Comput. **C-30**, 225 (1981).

[17] C. T. Bartee, P. I. Schneider, IRE Trans. Inf. Theory **IT-8**, 17 (1962).

[18] B. Benjauthrit and I. Reed, IEEE Trans. Comput. **C-25**, 78 (1976).

[19] T. C. Bartee and P. I. Schneider, Inf. Control **6**, 79 (1963).

[20] E. R. Berlecamp, H. Rumsey, and G. Solomon, Inf. Control **10**, 553 (1967).

[21] I. N. Aleksandrov, R. I. Gaĭdamaka, N. M. Nikityuk *et al.*, Preprint R10-84-865, JINR, Dubna (1984) [in Russian].

[22] R. I. Gaidamaka and N. M. Nikityuk, Preprint E10-88-53, JINR, Dubna (1988).

[23] N. M. Nikityuk, Preprint E11-87-10, JINR, Dubna (1987).

[24] R. E. Blauhut, Proc. IEEE **73**, 30 (1985).

[25] B. Benjauthrit and I. Reed, IEEE Trans. Comput. **C-27**, 757 (1978).

[26] D. K. Pradhan, IEEE Trans. Comput. **C-27**, 239 (1978).

[27] T. C. Wesselkamper, IEEE Trans. Comput. **C-27**, 232 (1978).

[28] I. N. Aleksandrov, R. I. Gaφdamaka, N. M. Nikityuk *et al.*, Preprint R10-84-865, JINR, Dubna (1984) [in Russian].

[29] N. M. Nikityuk, Preprint R11-85-865, JINR, Dubna (1985) [in Russian].

[30] N. M. Nikityuk, Avt. svid. SSSR No. 1236457, G06 7/00, OI, 1986, No. 21, p. 199.

[31] N. M. Nikityuk, Preprint R11-87-54, JINR, Dubna (1987) [in Russian].

[32] N. M. Nikityuk, Preprint R11-88-852, JINR, Dubna (1988) [in Russian].

[33] M. Jonson, A. J. Lankford, S. Amendolia *et al.*, Preprint SLAC-PUB-4611, Stanford University, Palo Alto (1988).

[34] N. M. Nikityuk, in *Proc. of the First Intern. Joint Conf. of ISSAC-88 and AAECC-6*, Rome, 1988, edited by T. Mora (Springer-Verlag, 1988), Vol. 357, p. 324; Preprint E10-88-28, JINR, Dubna (1988).

[35] N. M. Nikityuk, Preprint R10-87-254, JINR, Dubna (1987) [in Russian].

[36] N. M. Nikityuk, Preprint E-89-362, JINR, Dubna (1989).

[37] J. L. Messy, IEEE Trans. Inf. Theory **IT-11**, 580 (1965).

[38] R. B. Banerji, Proc. IRE **49**, 1585 (1961).

[39] E. L. Blokh, Izv. Akad. Nauk SSSR Tekh. Kibern. No. 3, 30 (1964) [in Russian].

[40] F. Polkinghorn, IEEE Trans. Inf. Theory **IT-12**, 480 (1966).

[41] R. T. Chien, D. B. Cunningham, and I. B. Oldham, IEEE Trans. Inf. Theory **IT-15**, 329 (1969).

[42] H. Okano and H. Imai, IEEE Trans. Comput. **C-36**, 1165 (1987).

[43] N. M. Nikityuk, Preprint R10-88-853, JINR, Dubna (1988) [in Russian].

[44] N. M. Nikityuk, Preprint R10-89-16, JINR, Dubna (1989) [in Russian].

[45] E. R. Berlekamp, IEEE Trans. Inf. Theory **IT-11**, 577 (1965).

[46] S. L. Hurst, IEEE Trans. Comput. **C-30**, 986 (1981).

[47] R. T. Chien, IEEE Trans. Inf. Theory **IT-10**, 357 (1964).

[48] A. P. Jeavons, N. Fora, D. Lindberg *et al.*, IEEE Trans. Nucl. Sci. **NS-23**, 250 (1976).

[49] N. M. Nikityuk, Preprint R10-88-742, JINR, Dubna (1988) [in Russian].

[50] W. Wagner, IEEE Trans. Inf. Theory **IT-16**, 649 (1970).

[51] B. Elspas, IRE Trans. Inf. Theory **IT-8**, 30 (1962).

[52] B. Dierichx, Nucl. Instrum. Methods A **275**, 542 (1989).

[53] S. Parker, Nucl. Instrum. Methods A **275**, 494 (1989).

[54] N. M. Nikityuk, Preprint E10-91-1612, JINR, Dubna (1991).

[55] N. M. Nikityuk, Preprint R10-88-854, JINR, Dubna (1988) [in Russian].

[56] H. M. Shao, T. K. Truong, J. Leslie *et al.*, IEEE Trans. Comput. **C-34**, 393 (1985).

[57] N. M. Nikityuk, Preprint R10-87-266, JINR, Dubna (1987) [in Russian].

[58] P. Calingaert, J. Assoc. Comput. Mach. **8**, 186 (1961).

[59] N. M. Nikityuk, Avt. svid. SSSR No. 15446992, OI, 1990, No. 8, p. 245.

[60] N. M. Nikityuk, Preprint E10-91-567, JINR, Dubna (1991).

[61] W. H. Kautz and R. C. Singleton, IEEE Trans. Inf. Theory **IT-10**, 363 (1964).

[62] N. M. Nikityuk, Prib. Tekh. Eksp. No. 3, 59 (1986) [Instrum. Exp. Tech.].

[63] N. M. Nikityuk, R. A. Rukoyatkin, and A. L. Svetov, Prib. Tekh. Eksp. No. 1, 95 (1991) [Instrum. Exp. Tech.].

[64] N. M. Nikityuk, Prib. Tekh. Eksp. No. 3, 74 (1983) [Instrum. Exp. Tech.].

[65] Ya. A. Khetagurov and Yu. P. Rudnev, *Increasing the Reliability of Digital Devices by Redundant Coding Techniques* [in Russian] (Énergiya, Moscow, 1974).

Translated by Patricia A. Millard