MATHEMATICS OF COMPUTATION, VOLUME 35, NUMBER 152
OCTOBER 1980, PAGES 1353–1359

Arithmetic in a Finite Field

By Michael Willett *

Abstract. An algorithm for realizing finite field arithmetic is presented. The relationship between linear recursions and polynomial arithmetic (modulo a fixed polynomial) over Zp is exploited to reduce the storage and computation requirements of the algorithm. A primitive normal polynomial is used to simplify the calculation of multiplicative inverses.

1. Introduction. Finite fields have become popular as algebraic structures in which to embed finite sets, especially in information theory [1], [5]. By thinking of the objects under study as field elements, one can often reduce a difficult combinatorial or geometric problem to an algebraic one. An example of this is Peterson's algorithm for decoding an error-correcting code [5]. The problem is to determine the unknown *locations* of errors in a binary n-tuple that has been transmitted over a noisy channel. The set $\{1, 2, \ldots, n\}$ of possible error locations is embedded in a finite field in such a way that the error locations become roots of a known polynomial over that field. Thus, locating the errors reduces to factoring polynomials over finite fields.

Various techniques for implementing finite field arithmetic have been introduced in the literature [2], [3], [5], [6]. For large fields, these realizations differ markedly in terms of their computational complexity and storage requirements. Also, the introduction of mathematical concepts such as group characters tends to obscure the development unnecessarily.

The purpose of this note is to exhibit an algorithm for arithmetic in a finite field, which this author feels has near minimal storage and computational requirements. Abstractly, this algorithm and the others mentioned are equivalent in the sense that, up to isomorphism, *the* arithmetic in a finite field is polynomial arithmetic, modulo a fixed polynomial. The differences lie in the interpretations given to the parameters involved.

2. Structure of Finite Fields. Let Zp denote the field of integers modulo p, where p is a prime number. This field is usually represented by the set $\{0, 1, 2, \ldots, p-1\}$ together with integer addition and multiplication modulo p. Let

$$f(x) = x^N - a_1 x^{N-1} - \cdots - a_N, \qquad a_N \neq 0, a_i \in Zp, \tag{1}$$

be a monic polynomial over Zp. Similar to the modulo p arithmetic for Zp, modulo $f(x)$

Received December 21, 1978.
1980 *Mathematics Subject Classification.* Primary 12C15; Secondary 12C10, 12C05.
*Work was completed while the author was a Visiting Associate Professor in the Department of Applied Mathematics and Computer Science at the University of Virginia.

0025-5718/80/0000-0177/$02.75

arithmetic can be performed on the set $Zp[x]$ of all polynomials over Zp. Since a division algorithm holds for $Zp[x]$, just as for the integers, we can define modulo $f(x)$ arithmetic by identifying each sum and product with the unique remainder after division by $f(x)$. The set $F = \{r(x) = b_0 + b_1x + \cdots + b_{N-1}x^{N-1} | b_i \in Zp\}$ of all possible remainders modulo $f(x)$ serves as a set of representatives for modulo $f(x)$ arithmetic. If $f(x)$ is *irreducible* over Zp, then F is a finite field with $|F| = p^N$ elements. Furthermore, for each degree N, irreducible polynomials $f(x)$ do exist so that finite fields of order p^N exist. Conversely, any finite field must have order which is a power of a prime and be isomorphic to the field constructed above. All finite fields of the same order are isomorphic so that any implementation of field arithmetic is implicitly equivalent to modulo $f(x)$ arithmetic.

The most important structural property of any finite field is that the multiplicative group of nonzero elements is cyclic, that is, there exists a primitive element $\theta \in F$ that generates the nonzero elements. If the polynomial $f(x)$ (irreducible over Zp) has a primitive element θ as a root over F, then $f(x)$ is called a *primitive polynomial.* There are $\phi(p^N - 1)/N$ primitive polynomials of degree N over Zp, where ϕ is Euler's function. Since $f(\theta) = 0$ implies that $\theta^N = a_1\theta^{N-1} + \cdots + a_N$, all powers of θ (and therefore all nonzero field elements) can be written uniquely as a linear combination over Zp of the set $\{1, \theta, \theta^2, \ldots, \theta^{N-1}\}$. In this way, F is viewed as an N-dimensional vector space over its subfield Zp.

TABLE 1

		1	θ	θ^2		θ^{N-1}
θ^0	=	$u_0^{(0)}$	$u_0^{(1)}$	$u_0^{(2)}$	$\cdots\cdots$	$u_0^{(N-1)}$
θ^1	=	$u_1^{(0)}$	$u_1^{(1)}$	$u_1^{(2)}$	$\cdots\cdots$	$u_1^{(N-1)}$
	.					
	.					
	.					
θ^n	=	$u_n^{(0)}$	$u_n^{(1)}$	$u_n^{(2)}$	$\cdots\cdots$	$u_n^{(N-1)}$
	.					
	.					
	.					
θ^{p^N-2}	=	$u_{p^N-2}^{(0)}$	$\cdots\cdots\cdots\cdots$			$u_{p^N-2}^{(N-1)}$

For a primitive polynomial $f(x)$ of degree N with primitive root θ, all the roots of $f(x)$ are given by $\theta, \theta^p, \theta^{p^2}, \ldots, \theta^{p^{N-1}}$, called the *conjugate set* for θ. If the conjugate set is linearly independent over Zp, the polynomial $f(x)$ is called a primitive *normal* polynomial. Davenport has shown that there exists at least $(p-1)^N$ such polynomials for each N and prime p. For the remainder of this paper, we will *assume that $f(x)$ is normal.*

Form the power table for θ (in Table 1), where the $u_i^{(j)} \in Zp$ and $u_i^{(j)} = \delta_{ij}$, $u_N^{(j)} = a_{N-j}$ for $0 \leqslant i, j \leqslant N-1$.

The right side of the power table consists of all the coordinates over Zp of the nonzero field elements in the standard basis $\{1, \theta, \theta^2, \ldots, \theta^{N-1}\}$. For small fields and/or hand calculations, the power table embodies both addition (coordinate representation, right side) and multiplication (exponent representation, left side). However, the table lookup required to convert between exponent and coordinate forms is time-consuming.

The power table has the additional property that each column $(u_0^{(i)}, u_1^{(i)}, u_2^{(i)}, \ldots)^T$ of coordinates, $0 \leqslant i \leqslant N-1$, satisfies the linear recursion

$$u_{n+N} = a_1 u_{n+N-1} + a_2 u_{n+N-2} + \cdots + a_N u_n, \qquad n = 0, 1, 2, \ldots, \tag{2}$$

which makes the power table easy to calculate, starting from the identity matrix at the top. Since (2) generates maximal length sequences called *m-sequences,* the columns of the power table are all cyclic shifts of one another.

As an example, let $f(x) = x^3 + x^2 + 1$ over the integers modulo 2. Then $f(x)$ is a primitive normal polynomial with resulting power table given in Table 2.

TABLE 2

	1	θ	θ^2
$\theta^0 =$	1	0	0
$\theta^1 =$	0	1	0
$\theta^2 =$	0	0	1
$\theta^3 =$	1	0	1
$\theta^4 =$	1	1	1
$\theta^5 =$	1	1	0
$\theta^6 =$	0	1	1

3. Coordinate Representation for Field Elements. Certain applications of finite fields, such as range-finding radar, may require very large fields. Orders of 2^N, $N > 50$, are encountered in information theory. It would be impossible to store the m-sequences of such lengths. Even if one used recursion (2) to generate the m-sequence each time a conversion between exponent and N-tuple was required, the time factor would be prohibitive. To build a special-purpose device to perform finite field arithmetic, one must find a field representation other than the power table (see Bartee and Schneider [2]).

Choose any basis $B = \{w_1, w_2, \ldots, w_N\} \subseteq F$ of F over Zp. Elements in F will be represented exclusively by their coordinates in this basis so that $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ when α is the field element $\Sigma \alpha_i w_i$. Then field addition is N-tuple addition as usual. Let $P_k(\alpha, \beta)$ for $\alpha, \beta \in F$ be the kth coordinate of the product of α, β. If $\alpha \cdot \beta = (c_1, c_2, \ldots, c_N)$, then $P_k(\alpha, \beta) = c_k$. The function P_k is linear in each argument so P_k is a bilinear functional from $F \times F$ into Zp. There exists an

$N \times N$ matrix M_k over Zp so that

$$P_k((\alpha_1, \alpha_2, \ldots, \alpha_N), (\beta_1, \beta_2, \ldots, \beta_N)) = (\alpha_1, \alpha_2, \ldots, \alpha_N) M_k (\beta_1, \beta_2, \ldots, \beta_N)^T.$$

In fact, the (i, j) entry of M_k is $P_k(w_i, w_j)$.

Assume that we are using the standard basis $w_i = \theta^{i-1}$ for $1 \leqslant i \leqslant N$. Then $P_k(w_i, w_j) = P_k(\theta^{i-1}, \theta^{j-1}) =$ the kth coordinate of θ^{i+j-2} in the standard basis. But this is merely the kth entry in the row of the power table (Table 1) corresponding to θ^{i+j-2} or $P_k(w_i, w_j) = u_{i+j-2}^{(k-1)}$. For the example in Table 2 we have

$$M_1 = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{vmatrix}, \quad M_2 = \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix}, \quad M_3 = \begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}.$$

For $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_N)$ write out the value of $P_k(\alpha, \beta)$.

$$(3) \qquad P_k(\alpha, \beta) = \alpha M_k \beta^T = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \beta_j u_{i+j-2}^{(k-1)} = \sum_{n=0}^{2N-2} \Big(\sum_{i+j=n+2} \alpha_i \beta_j \Big) u_n^{(k-1)}.$$

Since the columns of the power table satisfy recursion (2), the device shown schematically in Figure 1 (based on expression (3)) will calculate the product of two field elements *without* storing the matrices M_k. Expression (3) demonstrates explicitly how convoluted inner products and linear recursions are involved in the field representation.

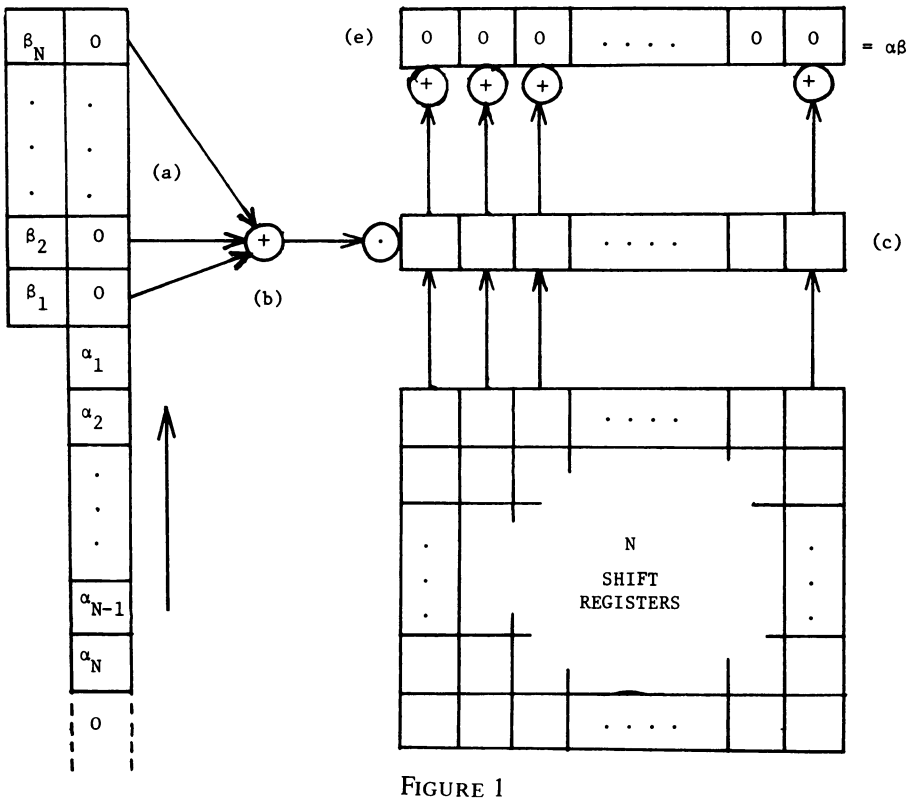

FIGURE 1

Initially, the coordinates of α and β are loaded as shown and the N vertical shift registers, each of which is wired (or programmed) to implement recursion (2), are loaded with the $N \times N$ identity matrix and the storage areas at (a) and (e) are loaded with zeros. After each time unit $n = 0, 1, \ldots, 2N - 2$ the following occurs:

(1) The contents of the $2N$ storage units at (a) are shifted up one space. The top entry is discarded and a zero is inserted at the bottom.

(2) Corresponding elements in the α and β storage units at (a) are multiplied and these products are summed at (b). Notice that the nontrivial overlap at (a) consists of all products $\alpha_i\beta_j$ for which $i + j = n + 2$.

(3) The N shift registers are activated so that the top row goes into storage unit (c) and the bottom row becomes the next terms generated by (2). Note that at time n, these registers will contain $(C(f))^n$, where

$$C(f) = \begin{vmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_N & a_{N-1} & & \cdots & a_1 \end{vmatrix}$$

is the companion matrix of the polynomial $f(x)$.

(4) Every element in storage (c) is multiplied by the sum from (b).

(5) The results are added to the contents of the storage units at (c).

(6) After $n = 2N - 2$, the storage unit (e) will contain the product of α and β.

This implementation demonstrates explicitly how the underlying modulo $f(x)$ arithmetic is performed using shift registers (linear recursions). The bank of shift registers is easily simulated in software.

4. Inverses in Large Fields. We have not said very much about division in a finite field. Bartee and Schneider [2] demonstrate how to determine a nonlinear circuit to realize division, which will not be discussed here.

For $\alpha \neq 0$, $\alpha \cdot \alpha^{p^N-2} = \alpha^{p^N-1} = 1$ so $\alpha^{-1} = \alpha^{p^N-2}$. To multiply by α^{-1} (i.e., to divide by α) one could multiply by α^{p^N-2}. Positive powers of a field element can be found by repeated multiplications by α. However, for large $p^N - 2$ this could be very time-consuming. Observe that $p^N - 2 = (\Sigma_{i=1}^{N-1}(p-1)p^i) + (p-2)$ so that if $\beta = \alpha^{p-1}$, then

$$\alpha^{p^N-2} = \alpha^{p-2}\beta^{p(1+p(\cdots+p(1+p))\cdots)} = \alpha^{p-2}(\cdots(((\beta^p)\beta)^p\beta)^p\cdots)^p,$$

where the exponent of β is $\Sigma_{i=1}^{N-1}p^i$. To form α^{p^N-2} we would first calculate β. Let $x_1 = \beta, x_2 = \beta^p\beta, \ldots, x_{i+1} = x_i^p\beta$. Then $\alpha^{-1} = \alpha^{p^N-2} = \alpha^{p-2}(x_{N-1})^p$. If N is large, then most of the time is spent raising successive x_i to the pth power. For large p and calculations involving numerous inversions, this could be quite slow.

Since $f(x)$ is assumed to be a primitive normal polynomial, the conjugate set

$\{\theta, \theta^p, \ldots, \theta^{p^{N-1}}\}$ can also be used as a basis for F over Zp. If $\beta = (\beta_1, \beta_2, \ldots, \beta_N)$ corresponds to $\Sigma_{i=1}^{N} \beta_i \theta^{p^{i-1}}$, then β^p corresponds to $(\beta_N, \beta_1, \beta_2, \ldots, \beta_{N-1})$ in the conjugate basis. The pth powers of an element can be found by merely shifting the coordinates cyclically to the right. In this basis, α^{-1} can be calculated by alternating coordinate shifts with multiplication by β to produce x_{N-1}.

The main problem with using the conjugate basis exclusively is that the shift register circuit in Figure 1 cannot be used for multiplication. One would have to store the N matrices M_k which represent the bilinear functionals associated with multiplication. The (i, j) entry in M_k is the kth coordinate of $\theta^{p^{i-1}+p^{j-1}}$ in the conjugate basis. The bilinear functionals do not seem to have a direct shift register implementation. A satisfactory compromise would be to represent field elements by their coordinates in the standard basis but change bases internally for the purpose of forming pth powers.

Let M be the $N \times N$ matrix over Zp which determines the change from the standard basis to the conjugate basis.

$$M \begin{vmatrix} 1 \\ \theta \\ \theta^2 \\ \cdot \\ \cdot \\ \cdot \\ \theta^{N-1} \end{vmatrix} = \begin{vmatrix} \theta \\ \theta^p \\ \theta^{p^2} \\ \cdot \\ \cdot \\ \cdot \\ \theta^{p^{N-1}} \end{vmatrix}$$

The ith row of M is merely the row in the power table (Table 1) corresponding to $\theta^{p^{i-1}}$. Invertibility of M is a test that $f(x)$ is in fact a primitive *normal* polynomial.

A modified power table (Table 3) could be formed in which each nonzero element is represented in the conjugate basis.

TABLE 3

		θ θ^p θ^{p^2} $\cdots$ $\theta^{p^{N-1}}$
1	=	
θ	=	
θ^2	=	M^{-1}
.		
.		
.		
θ^{N-1}	=	. - - - - - - - - - -
.		
.		. . . . .
.		
$\theta^{p^{N-2}}$	=	. . . .

The columns of Table 3 also satisfy recursion (2), with the initial N-tuples given by the columns of M^{-1}. If $c = (c_1, c_2, \ldots, c_N)$ are the coordinates of a field element in the standard basis, then cM^{-1} are the coordinates in the conjugate basis.

Observe that a right cyclic shift of an N-tuple c is accomplished by the multiplication cJ, where

$$J = \begin{vmatrix} & & \\ 0 & \cdots & I_{N-1} \\ & & \\ \cdots & \cdots & \cdots \\ 1 & \cdots & 0 \end{vmatrix}.$$

Let $c = (c_1, c_2, \ldots, c_N)$ be the coordinates of a field element in the standard basis. Then cM^{-1} are the coordinates in the conjugate basis and $cM^{-1}J$ is the right cyclic shift of these coordinates corresponding to raising c to the pth power. The N-tuple $cM^{-1}JM$ is then the coordinates of c^p in the standard basis. Multiplication by the matrix $M^{-1}JM$ corresponds to finding the pth power of an element in the standard basis directly. Software (or circuitry) could be added to Figure 1 to implement this multiplication, which would facilitate the calculation of field inverses.

4. Conclusions. All algorithms for realizing finite field arithmetic are equivalent to implementing polynomial arithmetic modulo a fixed polynomial over Zp. These realizations differ in the interpretation given to the parameters involved and in their storage and computation requirements. In this note we have presented one derivation of the explicit relationship between linear recursions and field multiplication. A primitive normal polynomial can be used to simplify the calculation of multiplicative inverses.

The algorithm presented in this note for finite field arithmetic only requires the storage of one primitive normal polynomial and the matrix $M^{-1}JM$ for use in calculating pth powers.

Department of Mathematics
University of North Carolina at Greensboro
Greensboro, North Carolina 27412

1. N. ABRAMSON, *Information Theory and Coding*, McGraw-Hill, New York, 1963.
2. T. C. BARTEE & D. I. SCHNEIDER, "Computation with finite fields," *Inform. and Control.*, v. 6, 1963, pp. 79–98.
3. J. T. B. BEARD, "Computing in GF(q)," *Math. Comp.*, v. 28, 1974, pp. 1159–1168.
4. H. DAVENPORT, "Bases for finite fields," *J. London Math. Soc.*, v. 43, 1968, pp. 21–39.
5. W. W. PETERSON & E. J. WELDON, Jr., *Error-Correcting Codes*, 2nd ed., M.I.T. Press, Cambridge, Mass., 1972.
6. G. R. REDINBO, *Finite Field Arithmetic on an Array Processor*, Electrical and Systems Engineering Department, Rensselaer Polytechnic Institute. (Unpublished.)