NICE du 16 au 20 MAI 1983

ALGORITHMS AND STRUCTURES FOR CONVOLUTIONS OVER GALOIS FIELDS

I.Pitas, M.G.Strintzis

Department of Electrical Engineering, University of Thessaloniki, Thessaloniki, GREECE

RESUME

Résumé

Le calcul des convolutions cycliques dans des Corps de Galois est une partie intégrante aussi bien de la théorie et de la formulation de Codage, que de nombreuses applications de Traitement du Signal. Dans le travail présent nous introduisons une méthode pour le calcul des convolutions de ce genre, qui minimise, en théorie, la complexité des calculs de l'algorithme. Nous proposons également des structures d' ordinateur pour la réalisation efficace de l' algorithme, et en général pour le calcul efficace des convolutions dans des Corps de Galois.

SUMMARY

Abstract

The computation of cyclic convolutions in Galois Fields is an integral part of Coding Theory and Formulation as well as of many signal Processing applications. In this paper, we introduce a method for the computation of such convolutions that minimizes, in theory, the computational complexity of the algorithm. We also propose special-purpose computer architecture schemes for the efficient realization of the algorithm, and in general for efficient calculation of convolutions in Galois Fields.

I. Introduction

As is well known [1-5] both cyclic and non-cyclic convolutions in Galois Fields are instrumental for the solution of several Signal Processing [2,9] and many Coding and Decoding Problems [5,6]. For instance cyclic convolutions in Galois Fields are needed for the decoding of among others, the BCH (Bose-Chaudhuri-Hohequem) and the Reed-Solomon codes [5,6].

In the following, GF(p) (where p is a prime integer) will be understood to denote the Galois Field with elements $\{0,1,..p-1\}$, and $GF(p^n) \equiv GF(p,f_n(x))$ the Galois Field generated by the n^{th} degree-polynomial $f_n(x)$ and composed of all polynomials with degree no greater than n and coefficients in GF(p). In $GF(p^n)$ multiplication of its element polynomials is defined modulo $f_n(x)$ and addition of the coefficients of the polynomials is defined modulo p.

Our purpose is the computation of the cyclic convolution of two sequences $h(\cdot)$ and $u(\cdot)$ belonging to $GF(p^n)$:

$$y(k) = \sum_{i=0}^{N-1} h(i)\ u(k-i) \qquad (1)$$

where k,i and k-i are understood to be computed modulo N. If Y(z), H(z) and U(z) are the z-transforms of, respectively, the finite sequences $\{y(0),..,y(N-1)\}$, $\{h(0),..,h(N-1)\}$ and $\{u(0),..,u(N-1)\}$, then the above is equivalent [7,8] to the computation of

$$Y(z) = U(z)\ H(z) \quad \mathrm{mod}\ (z^N-1) \qquad (2)$$

Since multiplications are particularly toilsome in Galois Fields, the computational complexity of (1) is almost entirely dependent on the required number of multiplications. It has been shown [7,8,10], that if p is nota factor of N and if

$$z^N-1 = \prod_{i=1}^{K} c_i(z), \quad \gcd(c_i(z), c_j(z)) = 1 \quad (3)$$

and the factors $c_i(z)$ are mutually prime and irreducible in $GF(p^n)$, then the minimum number of multiplications is

$$M = 2N-K \quad (4)$$

and the computation of (1) may be described by

$$\underline{y} = C[A\ \underline{u} \otimes B\ \underline{h}] \quad (5)$$

where $\otimes$ denotes Kronecker multiplication, $\underline{y}$, $\underline{u}$ and $\underline{h}$ are vectors composed of the elements of respectively the sequences $\{y(i)\}$, $\{u(i)\}$, $\{h(i)\}$, $i=0,1,..,N-1$, and A,B,C matrices of respective dimensions NxM, NxM and MxN, fully determined by the factorization (3). Thus, the minimum number of multiplications is reached when K is maximum.

Algorithms that achieve high values of K and thus efficient computation have been constructed in [6] and [11] for convolutions in the field $GF(2^n)$ by using the above methodology. A different methodology led , in [12] to efficient algorithms in the general case $GF(p^n)$. The results of [6,11] and [12] coincide if p=2. However, none of these algorithms achieves the theoretically minimum number of multiplications of (4). In this paper, we present a novel computational method in which the minimum number of multiplications is achieved, at the cost however, of the requirement of special-purpose software or hardware for its use. The method is presented in Section II. Proposed special-purpose Computer architectures for the realization of convolution are in Section III. Finally, examples of the application of the method are given in Section IV.

II. Description of the Algorithm

In the following we shall use the notation a/b to indicate that a divides b and a∤b to indicate the opposite. We first cite without proof the following known theorem [1].

Theorem 1

Suppose that p∤N. Then

(a) If N/p^n-1, the polynomial z^N-1 is fully reducible in $GF(p^n)$ and thus the number of irreducible factors of z^N-1 in $GF(p^n)$ is K=N.

(b) If $N\nmid p^n-1$, there exists a minimum integer e such that $N/p^{ne}-1$.

In the second case, z^N-1 is fully reducible in $GF(p^{ne})$

$$z^N-1 = \prod_i (z-a^i) \quad (6)$$

(a is root of unity in $GF(p^n)$).

To determine the factors $c_i(z)$ in (3) it now suffices to group the factors in (6) so as to achieve irreducibity of the product of each group in $GF(p^n)$. This is easily accomplished by the following procedure [1]. Let S be the set of all indices i in (6):

$$S = \{i = j(p^{ne}-1)/N \quad , 0\leqslant j\leqslant N-1\} = \cup S_i$$

where each S_i , i=1,2,.. is composed of the numbers i, ip^n, ip^{2n},..,$ip^{(N-1)n}$, calculated modulo $(p^{ne}-1)$. It can be seen [1] that the subsets S_i are disjoint: $S_i \cap S_j = \emptyset$ for $i\neq j$ and that its union equals S. Each $c_i(z)$ in (3) is then determined by the product of all factors in (6) with indices in the same S_i. Thus, K is the number of the subsets S_i of S. Following the determination of $\{c_i(z)\}$, the following simple steps lead to the realization (5) of the optimum algorithm:

(1) Determination of polynomials $R_i(z)$ such that

$$R_i(z) = \delta_{ij} \bmod c_j(z) \qquad 1\leqslant i\leqslant K \quad (7)$$

where δ_{ij} is the Kronecker delta.

(2) Evaluation of

$$H_i(z) = H(z) \bmod c_i(z), \qquad 1\leqslant i\leqslant K \quad (8)$$

$$U_i(z) = U(z) \bmod c_i(z), \qquad 1\leqslant i\leqslant K \quad (9)$$

(3) Evaluation of

$$Y_i(z) = H_i(z)\ U_i(z) \bmod c_i(z), \quad 1\leqslant i\leqslant K \quad (10)$$

(4) Reconstruction of Y(z) (using the Chinese Remainder Theorem [7])

$$Y(z) = \sum_{i=1}^{K} Y_i(z)\ R_i(z) \bmod z^N-1 \quad (11)$$

which correspond precisely to the form (5). For instance, the matrix operations involving A and B in (5) correspond respectively to the operations (8) and (9), while the multiplication by C in (5) corresponds to (10).

The optimality of the algorithm was achieved by adhering fully to the general scheme of [7,8,10] and thus factoring in (3) z^N-1 on $GF(p^n)$. By contrast the non-optimal methods in [6,11,12] are based on factoring z^N-1 on GF(p). The practical difficulty of applying the present "optimal" method stems from the same distinction.

Specifically, with the present method the matrices A,B,C in (5) are in $GF(p^n)$ hence are polynomials in x, while in [6,11,12] these matrices are composed of numbers in GF(p). Clearly then, efficient application of the present algorithm requires special software or hardware.

III. Architecture for computing sums in Galois Fields

We consider the sum

$$a(m) = \sum_{k=0}^{N-1} b(k)c(k,m) \bmod f_n(x),p;\ a,b,c\in GF(p) \quad (12)$$

Clearly

$$b(k) = \sum_{i=0}^{n-1} b(k,i)\, x^i;\ b(k,i)\in GF(p) \quad (13)$$

$$c(k,m) = \sum_{j=0}^{n-1} c(k,m,j)x^j,\ c(k,m,j)\in GF(p) \quad (14)$$

From (12,13,14) we obtain

$$a(m) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} d(i,j,m)x^{i+j} \bmod f_n(x) \quad (15)$$

where

$$d(i,j,m) = \sum_{k=0}^{N-1} b(k,i)\ c\ (k,m,j) \bmod p \quad (16)$$

Obviously (16) in its entirety and the reduction of (15) modulo $f_n(x)$ may be performed by using Read-Only-Memories (ROM's). The remaining simple additions for the evaluation of (15) may be performed either again by the use of a ROM or by a simple arithmetic unit. Another, possibly advantageous formulation of these equations is given by the combination of

$$a(m) = \sum_{j=0}^{n-1} f(j,m) \bmod p \quad (17)$$

$$f(j,m) = \sum_{k=0}^{N-1} b(k)\ c\ (k,m,j)x^j \bmod f_n(x) \quad (18)$$

Again, (18) is realized by the use of a ROM while (17) reduces to the summation of "shifts" by j of the sequence f(j,m). With either formulation we are led to architecture similar to that used in [19,20,18] for the computation of convolutions. Not surprisinly, since (12) may represent or by interpreted as a convolution. In fact, if this "convolution" is invariant, i.e if c(k,m)=c(m-k) the architecture is simplified:

$$a(m) = \sum_{j=0}^{n-1} f(j,m) \bmod p \quad (17')$$

$$f(j,m) = \sum_{k=0}^{N-1} b(k)\ c\ (m-k,j)x^j \bmod f_n(x) \quad (18')$$

as shown in Fig.1. We note that this architecture is very general and much more efficient than the classical architecture of Fig.2. For the special case of $GF(2^n)$ the architecture can be simplified even further since addition may be performed with simple EXOR gates and each clock in the ROM will process precisely one bit of each c(m-k). Thus, the computation time of each a(m) is reduced to n T_c where T_c is the clock period, a small fracture of the time required with the classical architecture of Fig.2. The form of the multiplier in $GF(2^4)$ is given in Fig.3.

IV. Examples

Let N=3, p=2, n=4. The generating polynomial of $GF(2^4)$ is $f_2(x) = x^4+x+1$. Since $3/(2^4-1)$, z^3-1 is fully reducible in $GF(2^4)$:

$z^3-1 = (z+1)(z+x^2+x)(z+x^2+x+1)$

Thus, $c_0(z)=z+1$, $c_1(z)=z+x^2+x$, $c_2(z)=z+x^2+x+1$ and from (7),

$R_0(z)=z^2+z+1$

$R_1(z)=(x^2+x)z^2+(x^2+x+1)z+1$

$R_2(z)=(x^2+x+1)z^2+(x^2+x)z+1$

Clearly (8) yields

$H_0=H(z) \bmod (z+1)=h_0+h_1+h_2$

$H_1=H(z) \bmod (z+x^2+x)=h_0+h_1(x^2+x)+h_2(x^2+x+1)$

$H_2=H(z) \bmod (z+x^2+x+1)=h_0+h_1(x^2+x+1)+h_2(x^2+x)$

Correspondingly,

$U_0=u_0+u_1+u_2$

$U_1=u_0+u_1(x^2+x)+u_2(x^2+x+1)$

$U_2=u_0+u_1(x^2+x+1)+u_2(x^2+x)$

following the multiplication

$$Y_0=U_0H_0,\quad Y_1=U_1H_1,\quad Y_2=U_2H_2 \quad (19)$$

the values of y_n are determined by using (11):

$y_0=Y_0+Y_1+Y_2$

$y_1=Y_0+Y_1(x^2+x+1)+Y_2(x^2+x)$

$y_2=Y_0+Y_1(x^2+x)+Y_2(x^2+x+1)$

Thus, the convolution requires only the 3 multiplications in (19) in keeping with the theoretical minimum (4), while the method in [6] requires 4 multiplications. An important difference however, is that the method in [6] requires no multiplication by power of x, i.e. no bit-by-bit calculations, while the present method does.

As a second example, let N=5, p=2 and n=2. As known, the generating polynomial of $GF(2^2)$ is $f_2(x)=x^2+x+1$. Clearly, $5 \nmid (2^2-1)$. However with e=2, $5/2^{2e}-1$ since $2^{2e}-1=15$. Thus, if a is the first root of $GF(2^4)$, we obtain

$$z^5-1 = (z-1)(z-a^3)(z-a^{12})(z-a^6)(z-a^9) \qquad (20)$$

Following the procedure outlined in Section II, we find the following factorization of z^5-1 in $GF(2^2)$:

$$z^5-1 = (z-1)(z^2+(x+1)z+1)(z^2+xz+1)$$

Thus K=3 and the total number of multiplications will be 2·5-3=7, while the algorithm in [6] requires 10 multiplications.

References

[1] N.J.Sloane, F.J.McWilliams "The Theory of Error Correcting Codes" North Holland, 1978

[2] G.R.Redinbo, B.O.Carhoun, B.L.Johnson "Fast algorithms for signal processing Using finite field operations" Proc.1982 IEEE Int.Conf.on ASSP, Paris

[3] N.S.Szabo, R.I.Tanaka "Residue Arithmetic and its applications to Computer technology" McGraw Hill, 1967.

[4] R.G.Gallager "Information theory and reliable communication" Wiley, 1968.

[5] R.E.Blahut "Theory and practice of error control codes" to be published in 1983 by Addison-Wesley.

[6] I.S.Reed et al. "Fast transforms for decoding Reed-Solomon Codes" IEE Proc., vol.128, Pt.F, No.1, pp.9-14, Feb.1981.

[7] J.H.McClellan, C.M.Rader "Number Theory in Digital Signal Processing" Prentice-Hall, 1979.

[8] H.J.Nussbaumer "Fast Fourier Transform and convolution algorithms" Springer Verlag, 1981.

[9] W.K.Jenkins "Complex residue number arithmetic for high speed Signal Processing" Elect.Lett. , vol.16, No.17, pp.660-661, Aug.1980.

[10] S.Winograd "Some bilinear forms whose multiplicative complexity depends on the field of constants" Math.Sys.Theory, vol.10, pp.169-180, 1977.

[11] T.K.Truong, R.L.Miller, I.S.Reed "Fast technique for computing syndromes of B.C.H. and Reed-Solomon codes" Elect. Lett., vol.15, No.22, pp.720-721, Oct.79.

[12] M.D.Wagh, S.D.Morgera "Structured design method for convolutions over finite fields" to be published in IEEE Trans. on Inf.Theory

[13] M.D.Wagh, S.D.Morgera "On the multidimensional techniques for algorithms over finite fields" submitted for publication in IEEE Trans. on Inf.Theory.

[14] R.C.Agarwal, C.S.Burrus "Fast one-dimensional digital convolution by multidimensional techniques" IEEE Trans. on ASSP., vol. ASSP-22, No.1, pp.1-10, Feb.1974.

[15] H.J.Nussbaumer "Fast polynomial transform algorithms for digital convolution. IEEE Trans. on ASSP, vol.ASSP-28, pp.205-215, 1980.

[16] Y.C.Yenq "Digital convolution algorithm for pipelining multiprocessor Systems" IEEE Trans. on Computers, vol.C-30, No.12, pp.966-973, Dec.1981.

[17] S.Winograd "Arithmetic complexity of computations" soc. for Industrial and Applied Mathematics, 1980.

[18] C.S.Burrus, "Digital Filter structures described by distributed arithmetic", IEEE Trans. Circuits Syst. vol. CAs-24, pp.674-680, Dec.1977.

[19] A.Peled, B.Liu, "A new hardware realisation of digital filters" IEEE Trans. on ASSP, vol.ASSP-22, pp.456-462, Dec.1974.

[20] S.Chu, C.S.Burrus "A Prime Factor FFT Algorithm using distributed arithmetic" IEEE Trans. on ASSP,vol.ASSP-30,pp.217-227,Apr.1982

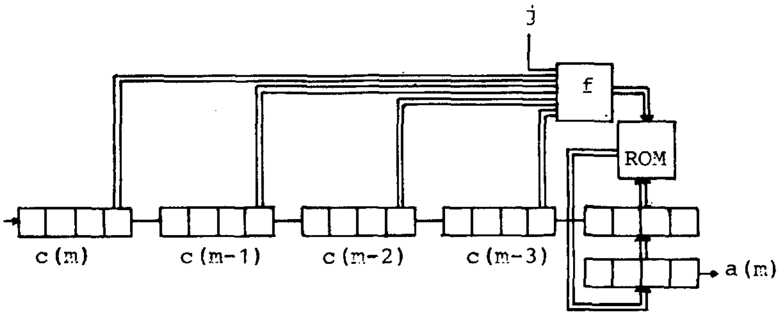

Fig.1: Proposed implementation of convolution.

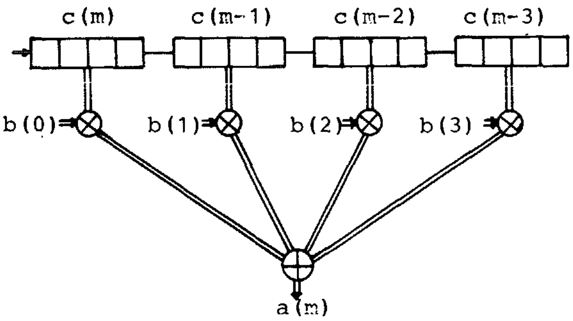

Fig.2:Classical implementation of convolution.

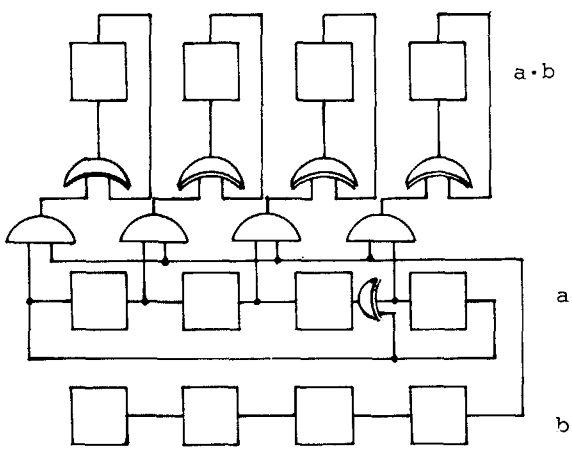

Fig.3:Circuit for multiplication over GF(2^4).