第 20 卷 第 6 期
1998 年 11 月

电 子 科 学 学 刊
JOURNAL OF ELECTRONICS

Vol.20 No.6
Nov. 1998

有限域 GF(2^m) 上的一个新的求逆算法 [1]

徐大专　　许宗泽

(南京航空航天大学电子工程系　南京　210016)

摘　要　根据有限域 GF(2^m) 上的正规基表示和 Massey-Omura 乘法器，本文提出了一个复杂性为 $O(\log m)$ 的求逆算法．新算法完成一次求逆运算只需要 $[\log_2(m-1)]+w(m-1)-1$ 次乘法和 $m-1$ 次循环移位，这里 $[x]$ 表示小于等于 x 的最大整数，$w(m-1)$ 表示 $m-1$ 的二进制表示中 "1" 的个数．

关键词　信息论，数字通信，算法，有限域，正规基，逆

中图号　TN911.2

1　引　言

求逆算法在编码理论和密码学中具有广泛的应用．例如，BCH 码的 Forney 算法就涉及求逆运算 [1]．应用求逆算法还可以降低指数运算的复杂性 [2]．当有限域 GF(2^m) 中的元素用多项式基表示时，求逆的典型算法是推广的 Euclid 算法 [1]．Massey 和 Omura 注意到，当有限域的元素用正规基表示时，只要进行一次循环移位就能完成一次平方运算 [3]．基于 Massey-Omura 并行乘法器，文献 [3] 还给出了一个复杂性为 $O(m)$ 的求逆算法．1990 年，文献 [4] 在快速指数算法的基础上提出了一个求逆算法，其复杂性为 $O(\sqrt{m})$[4]．当 m 为 2 的整数幂时，Itoh 和 Tsujii[5] 提出了一个求逆算法，该算法需要 $2(\log_2 m-1)$ 次乘法和 $m-1$ 次循环移位，即复杂性为 $O(\log m)$．Asano[6] 与前两位作者合作，对一般的 m 提出了一个推广的算法，但该算法存在明显的不足．首先，推广算法完全依赖于对 m 的素因子分解．只有当 m 能分解为很多小素数之积时，推广算法才比较有效．在 m 等于 2 的整数幂时达到最佳．因此，推广算法的复杂性通常要大于文献 [5] 中的算法．当 m 为素数或含有较大的素因子时，推广算法的作用甚小．另外，推广算法需要在 GF(2^m) 的一系列子域上进行乘法运算，这为算法的实现带来一定的困难．以上原因使推广算法的应用受到了较大的限制．本文对一般的正整数 m，提出了求逆运算的一个迭代算法，该算法至多需要 $2[\log_2(m-1)]$ 次乘法和 $m-1$ 次循环移位，这里 $[x]$ 表示小于等于 x 的最大整数，即算法的复杂性为 $O(\log m)$．当 $m-1$ 等于 2 的整数幂时，本算法只需 $\log_2(m-1)$ 次乘法．显然，本算法所需的乘法次数小于现有的算法．由于本算法不需要在子域上进行乘法运算，并且迭代公式非常规则，无需存贮额外的中间变量，因此非常便于实现．

2　新的求逆算法

设 x 是 GF(2^m) 上的任一非零元素，那么

$$x^{2^m-1}=1, \tag{1}$$

因此

$$x^{-1}=x^{2^m-2}. \tag{2}$$

[1] 1997-03-03 收到，1998-04-02 定稿

显然，求逆运算是一种特殊的指数运算．下面考虑形如下式：

$$y = x^{2^n-1} \tag{3}$$

的指数运算，这里 x 和 y 均以正规基表示．那么，平方运算等效于一次循环移位．设乘法运算由 Massey-Omura 并行乘法器 [4] 来完成，那么，完成一次乘法需要 m 次循环移位．显然，当 $n = m-1$ 时，$x^{-1} = y^2$ ．

设 n 的二进制表示式为

$$n = 2^k + a_1 2^{k-1} + a_2 2^{k-2} + \cdots + a_{k-2}2^2 + a_{k-1}2 + a_k, \tag{4}$$

这里 $k = [\log_2 n]$ 表示 n 的二进制展开式中的最高次幂，$[x]$ 表示小于等于 x 的最大整数．$a_i (0 \le i \le k)$ 等于 0 或 1，并且 $a_0 = 1$ ．令

$$\begin{aligned}
&n_0 = a_0 = 1,\\
&n_1 = 2 + a_1,\\
&n_2 = 2^2 + a_1 2 + a_2,\\
&\quad\vdots\\
&n_{k-1} = 2^{k-1} + a_1 2^{k-2} + \cdots + a_{k-2}2 + a_{k-1},\\
&n_k = n.
\end{aligned} \tag{5}$$

那么，我们可以得到如下的递推关系

$$n_{i+1} = 2n_i + a_{i+1}, \quad 0 \le i \le k-1. \tag{6}$$

通常 n 是预先知道的，因此，以上一系列 n_i 也可以预先确定．将 (6) 式中的 k 个等式相加，并整理可得

$$n = n_k = \sum_{i=0}^{k-1} n_i + \sum_{i=1}^{k} a_i + 1. \tag{7}$$

现在，我们定义

$$x_i = x^{2^{n_i}-1}, \tag{8}$$

显然有 $x_0 = x, y = x_k$ ．现在我们考虑 x_i 的计算问题．对一般的 i ，当 $a_{i+1} = 0$ 时，

$$2^{n_{i+1}} - 1 = 2^{2n_i} - 1 = 2^{n_i} - 1 + 2^{n_i}(2^{n_i} - 1),$$

那么

$$x_{i+1} = x^{2^{n_{i+1}}-1} = x^{2^{n_i}-1}\left(x^{2^{n_i}-1}\right)^{2^{n_i}} = x_i x_i^{2^{n_i}}.$$

当 $a_{i+1} = 1$ 时，

$$2^{n_{i+1}} - 1 = 2^{2n_i+1} - 1 = 1 + 2[2^{n_i} - 1 + 2^{n_i}(2^{n_i} - 1)],$$

那么

$$x_{i+1} = x^{2^{n_i+1}-1} = x[x^{2^{n_i}-1}(x^{2^{n_i}-1})^{2^{n_i}}]^2.$$

综上所述，我们可以得到计算 (3) 式的一个递推公式：

$$x_{i+1} = \begin{cases} x_i x_i^{2^{n_i}}, & a_{i+1} = 0, \\ x(x_i x_i^{2^{n_i}})^2, & a_{i+1} = 1. \end{cases} \tag{9}$$

用 (9) 式计算 x_1 需要 $1+a_1$ 次乘法和 $1+a_1$ 次循环移位；用 (10) 式计算 x_{i+1} 需要 $1+a_{i+1}$ 次乘法和 n_i+a_{i+1} 次循环移位．因此，计算 y 共需

$$\text{乘法次数} = k + \sum_{i=1}^{k} a_i;$$

$$\begin{aligned}\text{循环移位次数} &= (1+a_1) + (n_1+a_2) + \cdots + (n_{k-1}+a_k) \\ &= \sum_{i=0}^{k-1} n_i + \sum_{i=1}^{k} a_i = n-1(\text{根据 (7) 式}).\end{aligned}$$

令 $w(n)$ 表示 n 的二进制表示中 "1" 的个数，显然， $w(n) \le k+1$ ．那么，计算 y 共需 $k+[w(n)-1]$ 次乘法，至多需要 $2k = [\log_2 n]$ 次乘法，即算法的复杂性为 $O(\log n)$ ．当 n 是 2 的整数幂 2^k 时， $w(n)=1$ ，所需的乘法次数最少，这时计算 y 只需要 $\log_2 n$ 次乘法．

以上所述可以归结为下面的算法：

新的求逆算法

第一步　初始化．令 $n = m-1$ ，根据 (4) 式和 (5) 式确定 k, a_i 和 n_i ．置 $i=1$ ；

第二步　由 (9) 式的迭代算法计算 x_i ；

第三步　$i \leftarrow i+1$ ，如果 $i \le k$ ，转第二步；

第四步　$x^{-1} = x_k^2$ ，结束．

容易看出，求逆算法需要 $[\log_2(m-1)] + w(m-1) - 1$ 次乘法和 $m-1$ 次循环移位．

现举例说明新的求逆算法的执行过程．设 $m=100$ ， x 是 GF(2^{100}) 上的任一非零元素．由 $m-1 = 2^6+2^5+2+1$ 可得 k=6, $a_0 = a_1 = 1$, $a_2 = a_3 = a_4 = 0$, $a_5 = a_6 = 1$; $n_0 = 1$, $n_1 = 3$, $n_2 = 6$, $n_3 = 12$, n_4=24, n_5=49 ．由 (9) 式的迭代公式得

$$\begin{aligned}
x_1 &= x(xx^2)^2 = x^{2^3-1}, \\
x_2 &= x_1 x_1^{2^3} = x^{2^6-1}, \\
x_3 &= x_2 x_2^{2^6} = x^{2^{12}-1}, \\
x_4 &= x_3 x_3^{2^{12}} = x^{2^{24}-1}, \\
x_5 &= x(x_4 x_4^{2^{24}})^2 = x(x^{2^{48}-1})^2 = x^{2^{49}-1}, \\
x_6 &= x(x_5 x_5^{2^{49}})^2 = x(x^{2^{98}-1})^2 = x^{2^{99}-1}, \\
x^{-1} &= x_6^2 = x^{2^{100}-2}.
\end{aligned}$$

总共需要 9 次乘法和 99 次循环移位．

3 结 论

根据有限域 GF(2^m) 上的正规基表示和 Massey-Omura 并行乘法器，本文提出了求逆运算的一个迭代算法，其复杂性为 $O(\log m)$. 与现有的求逆算法相比，新算法不仅所需的乘法次数最少，而且通用性强，容易实现. 由于有限域中的求逆运算是一种非常基本的运算，因此新算法在编码理论和密码学等领域中具有广阔的应用前景.

参 考 文 献

[1] Berlekamp E R. Algebraic Coding Theory. New york: McGraw-Hill, 1968.
[2] Brickell F F. A fast modular multiplication algorithm with application to two key cryptography, advances in cryptography. Proceedings of Crypto-82, New York: Plenum Press, 1983, 51–60.
[3] Wang C C, Truong T K, Shao H M, Deutsch L J, Omura J K, Reed I S. VLSI architectures for computing multiplications and inverses in GF(2^m). IEEE Trans. on Computers, 1985, C-34(8): 709–716.
[4] 徐大专. 在 GF(2^m) 上计算指数和逆. 计算机学报, 1990, 13(11): 860–863.
[5] Itoh T, Tsujii S. Effective recursive algorithm for computing multiplicative inverses in GF(2^m). Electron. Lett., 1988, 24(6): 334–335.
[6] Asano Y, Itoh T, Tsujii S. Generalised fast algorithm for computing multiplicative inverses in GF(2^m). Electron. Lett., 1989, 25(10): 664–665.

A NEW ALGORITHM FOR COMPUTING INVERSES IN THE FINITE FIELD GF(2^m)

Xu Dazhuan Xu Zongze

(*Dept. of Electron. Eng., Nanjing University of Aeronautics and Astronautics, Nanjing* 210016)

Abstract A new algorithm with the complexity $O(\log m)$ is presented to compute inverses in the finite field GF(2^m) based on the normal basis representations and the Massey-Omura's multipliers. The inverse in GF(2^m) can be computed with $[\log_2(m-1)] + w(m-1) - 1$ multiplications and $m-1$ cyclic shifts, where $[x]$ denotes the maximum integer less than or equal to x, $w(m-1)$ the number of "1" in the binary representation of $m-1$.

Key words Information theory, Digital communication, Algorithm, Finite field, Normal basis, Inverse

徐大专： 男，1963 年生，硕士，主要研究领域有：编码理论，密码学，通信系统理论，神经网络和计算机软硬件设计等.
许宗泽： 男，1940 年生，教授，长期从事无线通信方面的教学和科研工作.