磁盘阵列中高速并行 RS 译码算法研究 [1]

王福文 董燕琴 李 兵
(第二炮兵第四研究所 北京 100085)

摘 要 该文在总结研究 RS 译码的基础上，给出了一种适合并行方式进行高速 RS 译码的方法，该方法对于高速数据磁盘阵列录取系统、高速数据通信系统的纠错译码效果显著，已成功地应用到磁盘阵列高速数据录取系统中.

关键词 RS 译码，磁盘阵列
中图号 TN76

1 概 述

纠错编码已经发展了几十年，但把纠错码用于 RAID(Redundant Array of Inexpensive Disk) 系统却是一个很新的技术，因为 RAID 的发展只有十几年的时间. 对于有 n 个设备组成的存储系统，假定每个设备的平均失效时间为 F，则整个系统的平均失效时间为 F/n，因此必须考虑系统容错.

对于小的 n 值，一个校验设备用于容错就足够，如在 RAID 系统中的 RAID5，在编码技术中称为 "$(n+1)$-parity". 该编码中校验数据的第 i 个字节由每个数据盘中的第 i 个字节异或得到，这样对于 $n+1$ 个设备中的任意一个设备失效，都可以通过其它的 n 个设备数据异或操作进行恢复. $(n+1)$-parity 设备因其实现简单而得到广泛的应用，它的不足之处在于不能恢复多于两个设备的同时失效. 随着 n 值的增加，许多个设备同时失效变得非常重要. 相应的技术已有报道 [1−3]. 对于有 m 个校验设备的系统，能够同时恢复 m 个设备失效的技术是采用 RS(Reed-Solomon) 编码.

在 RS 编译码操作中，运算最复杂并最耗费时间的操作是译码过程. 而译码操作中运算最复杂的是有限域的除法 / 倒数操作. 传统的实现倒数或除法的硬件方法是采用查找表，对于小的 m 值，该方法相当有效. 由于该方法在硬件实现上占用面积量级为 $O(m \bullet 2^m)$, 因此对于大的 m 值该方法就不适合用硬件实现. 最近几年，为减小硬件实现面积而采用的方法介绍很多 [2,4−9], 大多数方法基于 Systolic 阵列或相似的思想来设计. 这些倒数电路可以采用正交基 [4,5,9]、标准基 [2,5] 和对偶基 [5−8]. 正交基和对偶基要进行基变换，而标准基则不需要. 文献 [4] 的方法采用并入串出的方法来实现，文献 [9] 则采用串入并出的方法来实现，文献 [6–8, 10] 采用串入串出的方法，上述 3 种方法的面积和时间复杂度分别为 $O(m^3)$, $O(m \bullet \log_2 m)$； $O(m)$, $O(m \bullet \log_2 m)$； $O(m^2)$, $O(m)$. 文献 [2] 采用并入并出的方法，其面积和时间的复杂度都为 $O(m)$. 该方法需要 $2m$ 个时钟产生一个数据，因此在高速应用场合不是很合适，但在算法硬件化方面却很好，因为其算法占用的硬件面积很小.

本文在解决求解有限域倒数或除法运算的问题上，根据 Brunner 等人给出的修正的 Euclid 多项式最大公约数 (GCD； Greatest Common Divisor) 算法，提出了对该算法的一种修正，使之适合并发操作. 同以前的算法相比，该算法采用并入并出的并行加流水的操作，该算法占用的器件面积复杂度为 $O(m^2)$，吞吐率为每个时钟一个数据结果，其面积时间积为 $O(m^2)$ 量级. 因此本方法更适合于用 Systolic 阵列实现，并为实现高速并发的 RS 译码提供算法基础.

本文的结构如下：第 2 节介绍有限域 GF(2^m) 中计算倒数和除法运算的修正 Euclid 算法，第 3 节提出便于并行处理的 RS 译码算法——对修正 Euclid 算法的进一步修正；第 4 节就本文提出的算法中的关键技术作进一步分析，最后就全文进行了小结.

[1] 2000-11-24 收到，2001-06-09 定稿

2 有限域 GF(2^m) 中计算倒数和除法运算的修正 Euclid 算法

RS 的译码过程参见文献 [10] 。本节在文献 [10] 介绍的译码方法基础上，重点解决有限域中倒数和除法运算的算法实现。

在有限域中计算倒数和除法运算的修正的 Euclid 算法如下：令

$$G(x) = x^m + g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_1x + g_0 \tag{1}$$

为 GF(2) 中的本元多项式， α 为它的一个根，则有下面的方程：

$$x^m = g_{m-1}\alpha^{m-1} + g_{m-2}\alpha^{m-2} + \cdots + g_1\alpha + g_0 \tag{2}$$

对于 α ，建立一个有 2^m 个元素 $\{0, 1, \alpha, \alpha^2, \cdots, \alpha^{2^m-2}\}$ 的有限域 GF(2^m) ，当采用标准基 $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ 时，该域中的每个元素可以表示为阶数小于 m 的多项式。令 $A(x)$ ， $B(x)$ 为 GF(2^m) 中的两个元素的多项式表达式， $C(x)$ 为 $A(x)/B(x)\bmod G(x)$ 的商，即

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0 \tag{3}$$

$$B(x) = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1x + b_0 \tag{4}$$

$$C(x) = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_1x + c_0 \tag{5}$$

上述多项式的系数属于集合 $\{0, 1\}$ ，所有操作的系数执行模 2 操作。当 $A(x) = 1$ 时， $C(x)$ 为 $B(x)$ 的倒数。

在文献 [2] 中， Brunner 等人 [3] 给出了一个修正的 Euclid 多项式 GCD 算法，其算法可归纳为

$R = B(x); S = G(x); U = 1; V = 0;$
count=0;
for $i = 1$ to $2m$ do
  if $r_m = 0$ then //(r_m 表示 R 中 x^m 项的系数)
    $R = x * R$; $U = x * U \bmod G(x)$;
    count=count+1;
  else
    if $s_m = 1$ then //(s_m 表示 S 中 x^m 项的系数)
      $S = S + R$; $V = V + U$;
    end
    $S = x * S$;
    If count==0 then
      $R \leftrightarrow S$; $U \leftrightarrow V$; //(交换操作)
      $U = x * U \bmod G(x)$;
      count=count+1;
  else
      $U = U/x \bmod G(x)$;
      count=count−1;
  end
end
end //(此时 U 的值为 $1/B(x)\bmod G(x)$ ， 计数值为 0)

上述算法需要 $2m$ 次迭代并且计数值在最后一次迭代后总是为 0 。换言之，语句“ count=count +1 ”和“ count=count-1 ”在 $2m$ 次迭代中各执行 m 次。为实现上述算法，文献 [2] 给出了一个并入并出的流水结构，该电路的面积时间积性能优良，但不适合于 Systolic 阵列的实现，并

且存在广播问题。其不适合 Systolic 阵列实现的原因是在 $2m$ 次迭代中其算术运算的不一致性所致，有时执行 $U = x * U \bmod G(x)$，有时执行 $U = U/x \bmod G(x)$。

3 对修正的 Euclid 算法的修正

在上述算法中我们注意到在计算除法运算时可以用初始状态 $U = A(x)$ 来代替 $U = 1$，当去掉 $U = U/x \bmod G(x)$ 语句时，结果将变成 $U = x^m/B(x) \bmod G(x)$ 而不是期望的 $U = 1/B(x) \times \bmod G(x)$，为得到最终结果，我们可以在 $2m$ 次迭代完成后再执行 m 次 $U = U/x \bmod G(x)$ 操作；同时，当去掉 $U = U/x \bmod G(x)$ 操作后等效于执行 $V = x * V \bmod G(x)$(* 表示乘法，下同)。另外，操作 $R \leftrightarrow S$; $U \leftrightarrow V$; $U = x * U \bmod G(x)$; 等效于 $V = x * V \bmod G(x)$；$R \leftrightarrow S$; $U \leftrightarrow V$；通过上述修正，有下面的倒数 / 除法操作：

$R = B(x); S = G(x); U = 1; V = 0;$
count=0;
part A：
for $i = 1$ to $2m$ do
　if $r_m = 0$ then //(r_m 表示 R 中 x^m 项的系数)
　　$R = x * R$; $U = x * U \bmod G(x)$; (Key operation I)
　　count=count+1;
　else
　　if $s_m = 1$ then)
　　$S = S + R$; $V = V + U$;
　　end
　　$S = x * S$; $V = x * V \bmod G(x)$; (Key operation II)
　　If count==0 then
　　$R \leftrightarrow S$; $U \leftrightarrow V$; //(交换操作)
　　$U = x * U \bmod G(x)$;
　　count=count+1;
　else
　　count=count−1;
　end
end
end
　part B:
　for $i = 2m + 1$ to $3m$ do
　　$U = U/x \bmod G(x)$; (Key operation III)
　End //(此时 U 的值为 $A(x)/B(x) \bmod G(x)$)

本算法包含两部分运算，第一部分运算产生了一个中间结果 $C(x) * x^m \bmod G(x)$，第二部分则通过除 x^m 从而得到最终的结果。上面的运算过程与文献 [2] 给出的算法相比，在迭代操作时包含了一致性运算，从而该算法更适合于 Systolic 阵列实现。

4 主要操作 (Key Operation) 的实现

上面的算法过程中，有 3 个主要操作需要解决，它涉及到算法是否能够实现并发操作。由于 R 和 S 是最大阶数为 m 的多项式，而 U 和 V 的最大阶数为 $m - 1$，因此他们可以表达如下：

$$R = r_m x^m + r_{m-1}x^{m-1} + \cdots + r_1 x + r_0 \tag{6}$$

$$S = s_m x^m + s_{m-1}x^{m-1} + \cdots + s_1 x + s_0 \tag{7}$$

$$U = u_{m-1}x^{m-1} + u_{m-2}x^{m-2} + \cdots + u_1 x + u_0 \tag{8}$$

$$V = v_{m-1}x^{m-1} + v_{m-2}x^{m-2} + \cdots + v_1 x + v_0 \tag{9}$$

(1) $R = x * R$ 操作

令

$$R' \equiv r'_m x^m + r'_{m-1}x^{m-1} + \cdots + r'_1 x + r'_0 \tag{10}$$

从 (6) 和 (10) 式及下面的事实： $R = x * R$ 仅在 $r_m = 0$ 的条件下才有可能，因此，

$$r'_0 = 0 \tag{11}$$

$$r'_i = r_{i-1}, \quad 1 \le i \le m \tag{12}$$

同理，对 $S = x * S$ 亦有相同的结论。

(2) $U = x * U \bmod G(x)$ 操作

令

$$U' \equiv u'_{m-1}x^{m-1} + u'_{m-2}x^{m-2} + \cdots + u'_1 x + u'_0 = x * U \mathrm{mod} G(x) \tag{13}$$

由 (17) 式，

$$x^m \mathrm{mod} G(x) = g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_1 x + g_0 \tag{14}$$

把 (8) 式代入 (13) 式，并根据 (14) 式可以推导出下面的关系式：

$$u'_0 = u_{m-1}g_0 \tag{15}$$

$$u'_i = u_{m-1}g_i + u_{i-1}, \quad 1 \le i \le m-1 \tag{16}$$

同样， $V = x * V \bmod G(x)$ 也可以采用同样的方法。

(3) $U = U/x \bmod G(x)$ 操作

由于 $g_0 = 1$ ，重写 (14) 式：

$$1 = (x^{m-1} + g_{m-1}x^{m-2} + g_{m-2}x^{m-3} + \cdots + g_2 x + g_1)x \mathrm{mod} G(x) \tag{17}$$

从上式中我们可以得出下面的公式：

$$x^{-1} \mathrm{mod} G(x) = (x^{m-1} + g_{m-1}x^{m-2} + g_{m-2}x^{m-3} + \cdots + g_2 x + g_1) \tag{18}$$

令

$$U' \equiv u'_{m-1}x^{m-1} + u'_{m-2}x^{m-2} + \cdots + u'_1 x + u'_0 = U/x \mathrm{mod} G(x) \tag{19}$$

把 (8) 和 (18) 式代入 (19) 式，可推导出下面的关系式：

$$u'_{m-1} = u_0 \tag{20}$$

$$u'_i = u_0 g_{i+1} + u_{i+1}, \quad 1 \le i \le m-2 \tag{21}$$

上述推导是 RS 译码操作有限域中倒数 / 除法运算的算法分解，通过上述分解，可以实现 RS 译码操作的并行加流水的硬件实现，从而提高译码操作的速度，推广其应用范围。

5 小 结

本文在研究了 RS 译码的常用方法后，根据修正的 Euclid 算法，提出了一种适合于并发操作的 RS 译码方法，该方法在提高译码速度、硬件实现的简洁性和低的资源占用等方面有明显的优势，便于 FPGA 和 ASIC 器件的实现。

参 考 文 献

[1] W. A. Burkhard, J. Menon, Disk array storage system reliability, In 23rd International Symposium on Fault-Tolerant Computing, Toulouse, France, June 1993, 432–441.

[2] G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, D. A. Patterson, Failure correction techniques for large disk arrays, In Third International Conference on Architectural Support for Programming Languages and Operating Systems, Boston, MA, Apr. 1989, 123–132.

[3] H. Brunner, A. Curiger, M. Hofstetter, On computing multiplicative inverses in GF(2^m), IEEE Trans. on Comput., 1989, 42(8), 1010–1015.

[4] C. C. Wang, T. K. Truong, K. wang, H. M. Shao, L. J. Deutsch, J. K. Omura, I. S. Reed, VLSI architectures for computing multiplication and inverse in, IEEE Trans. on Comput., 1985, C-34(8), 709–717.

[5] I. S. Hsu, T. K. Truong, L. J. Deutsch, I. S. Reed, A comparison of VLSI architecture of finite field multipliers using dual, normal, or standard bases, IEEE Trans. On Comput., 1988, 37(6), 735–739.

[6] S. T. J. Fenn, M. Benaissa, D. Taylor, multiplication and division over the dual basis, IEEE Trans. On Comput., 1996, 45(3), 319–327.

[7] S. T. J. Fenn, M. Benaissa, D. Taylor, Division in GF(2^m), Electron. Lett., 1992, 28(19), 2259–2261.

[8] S. T. J. Fenn, M. Benaissa, D. Taylor, Improved algorithm for division over, Electron. Lett., 1993, 29(4), 469–470.

[9] G.L. Feng, A VLSI Architecture for Fast Inversion in GF(2^m), IEEE Trans. on Comput., 1989, 38(10), 1383–1386.

[10] Kuang Yung Liu, Architecture for VLSI design of Reed-Solomon decoders, IEEE Trans. on Comput, 1984, C-33(2), 178–189.

RESEARCH ON HIGH SPEED AND PARALLEL RS DECODING ALGORITHM IN DISK ARRAY

Wang Fuwen　Dong Yanqin　Li Bing

(*The Forth Institute of the Second Artillery Corps, Beijing* 100085, *China*)

Abstract A method of high Speed RS decoding which is propitious to parallel implementation is presented in this paper. The method is suitable for error correcting decoding with high speed data recording system of disk array and high speed data communicating system. It is successfully used in high speed data recording system of disk array.

Key words Reed-Solomon decoding, Disk array

王福文：　男，1970 年生，工程师，主要研究方向为信号与信息处理、高速数据录取系统等.

董燕琴：　女，1964 年生，高级工程师，主要研究方向为制导控制技术.

李　兵：　男，1964 年生，工程师，主要研究方向为控制技术.