

BCH 码迭代译码算法及软件实现方法

李志国, 张伟功

(西安微电子技术研究所, 陕西 西安 710075)

摘 要:结合 BCH 码的特点, 重点研究了 BCH 码中 BM 迭代译码算法的基本原理, 对二进制 BCH 码与非二进制 BCH 码作了简单的比较, 给出了算法的关键代码。根据 BM 迭代译码算法的基本步骤, 采用三级流水算法结构并对实际应用的缩短码(50,32)(纠二检四)译码进行分析, 同时阐明如何应用 C/C++ 语言实现该算法。

关键词:BCH 码; 译码; 有限域; 缩短码; 算法

中图分类号:TP311.52

文献标识码:A

文章编号:1673-629X(2007)04-0171-04

Binary BCH Code BM Decoding Algorithm and Software Realization

LI Zhi-guo, ZHANG Wei-gong

(Xi'an Microelectronics Technology Institute, Xi'an 710075, China)

Abstract:Based on the characteristics of BCH code and comparing with the non-binary BCH code to the binary BCH code, the critical code of BM algorithm is presented with emphasis on the fundamentals of BM algorithm. Furthermore, according to the basic algorithm steps of BM, the practical shortened code (50,32) is analyzed through the three-stage pipeline algorithm architecture. Its realization by C/C++ is also put forward.

Key words:BCH code; decoding; Galois field; shortened code; algorithm

1 BCH 码简介

BCH(Bose - Chaudhuri - Hocquenghem)码是 1959 年由霍昆格姆(Hocquenghem), 1960 年由博斯(Bose)和雷·查德胡里(Ray Chaudhuri)提出的纠正多个随机错误的循环码, 可以用生成多项式 $g(x)$ 的根来描述。它是具有严格的代数结构, 纠错能力强、构造简单、编码较其它码容易等特点的线性分组码。

2 BCH 码的译码方法

关于 BCH 码的译码问题, 一直是编码理论研究中感兴趣的课题之一。评价一个码的译码算法的好坏, 往往取决于译码速度和实现的难易程度以及译码错误概率的大小, 尤其在中短码长下, BCH 码有很好的纠错性能, 构造容易, 所以在实际中得到广泛应用, 而且 BCH 码与其他各类码有极其密切的关系, 因此研究 BCH 码的译码算法在理论上和实际上都有着重要

意义。在所有的译码算法中常用的有: Peterson 算法、Berlekamp - Massey 迭代算法(简称 BM 法)、欧几里得(Euclid)算法。Peterson 算法适用纠错数较少的译码, 它的计算量在上述三种算法中最大; 对于纠错数较大的 RS 译码器, 一般采用 BM 算法和 Euclid 算法, Euclid 算法比 BM 迭代算法容易理解, 但运算量较大, 且比 BM 迭代算法的硬件结构复杂。另外考虑到二进制 BCH 码的特殊性, 所以在本设计中采用译码速度快而最常用的 BM 迭代算法^[1]。

在实际应用中用得最多的是码元取自有限域 $GF(2^m)$ 中的二进制 BCH 码, 对于一个正整数 m 则有: 取 $m_0 = 1, \delta = 2t + 1$, 设 α 是伽罗华域 $GF(2^m)$ 的本原域元素, 则码以 $\alpha, \alpha^2, \dots, \alpha^{2^t}$ 为根生成多项式:

$$g(x) = \text{LCM}(m_1(x)m_2(x)\cdots m_{2^t}(x))$$

式中 $m_i(x)$ 是 $\alpha^i (1 \leq i \leq 2t)$ 相应的最小多项式, t 为纠正错误个数。由于在特征为 2 的 $GF(2^m)$ 域上, α^{2^i} 的最小多项式与 α^i 的相同, 所以生成多项式简化为:

$$g(x) = m_1(x)m_3(x)\cdots m_{2^t-1}(x)$$

相应地, 二进制 BCH 码以 $\alpha, \alpha^3, \alpha^5, \dots, \alpha^{2^t-1}$ 为根, 码长 $n = \text{LCM}(\varphi_1, \varphi_3, \dots, \varphi_{2^t-1})$, 码的校验矩阵就为

收稿日期: 2006-07-11

作者简介: 李志国(1977-), 男, 重庆人, 硕士研究生, 研究方向为计算机应用技术; 张伟功, 博士, 研究员, 研究方向为计算机系统结构及容错技术。

$$H = \begin{bmatrix} \alpha^{n-1} & \alpha^{n-2} & \cdots & \alpha & 1 \\ (\alpha^3)^{n-1} & (\alpha^3)^{n-2} & \cdots & \alpha^3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (\alpha^{2t-1})^{n-1} & (\alpha^{2t-1})^{n-2} & \cdots & \alpha^{2t-1} & 1 \end{bmatrix}$$

其中 $\varphi_1, \varphi_3, \dots, \varphi_{2t-1}$ 分别是 $\alpha, \alpha^3, \alpha^5, \dots, \alpha^{2t-1}$ 对应元素的级, 所以二进制 BCH 码的码长是 $n \leq 2^m - 1$, 当 $n = 2^m - 1$ 时, 称为本原长码, 当 $n < 2^m - 1$ 时则称为缩短码(shortened code)^[2]。校验位数目至多为 $\partial_0 g(x) = mt$ 个, 设计最短距离 $d_{\min} = 2t + 1$, 可纠正 $\leq t$ 个随机错误。

2.1 编码算法

设输入信息多项式

$I(x) = I_0 + I_1x + \cdots + I_{k-1}x^{k-1}$, 校验多项式 $P(x) = P_0 + P_1x + P_2x^2 + \cdots + P_{n-k-1}x^{n-k-1}$, 则 $x^{2t}I(x)$ 对生成多项式 $g(x)$ 求模得到的余式就是校验多项式 $P(x)$, 即 $P(x) = x^{2t}I(x) \pmod{g(x)}$, 那么生成码子的多项式为 $C(x) = x^{2t}I(x) + P(x)$ 。所以 BCH 码编码的实质就是以生成多项式 $g(x)$ 为模的除法问题。通常采用 LFSR(线性反馈移位寄存器) 来实现。

2.2 译码算法

在 BM 译码算法的整个过程中, 其译码算法主要分以下三步:

第一步: 由接收到的 $R(x)$ 计算伴随式 $S = (s_1, s_2, \dots, s_{2t})$;

第二步: 根据伴随式 S 求错误位置多项式 $\sigma(x) = \sigma_t x^t + \sigma_{t-1} x^{t-1} + \cdots + \sigma_1 x + 1$;

第三步: 求解 $\sigma(x)$ 的根(钱氏(Chien)搜索法)确定错误位置, 并进行错误纠正。

因为 $R(x) = C(x) + E(x)$, 所以

$$s_j = \sum_{i=1}^t Y_i(\alpha^j) = R(\alpha^j) \quad j = 1, 2, \dots, 2t-1$$

$Y_i \in GF(2)$, $E(x)$ 为错误图样, 而由码的生成多项式可以得出 $R(x) = q_j(x)m_j(x) + r_j(x)$ 。式中 $m_j(x)$ 是 α 的最小多项式, 且 $m_j(x)$ 是以 α^j 为根, 因此当 $x = \alpha^j$ 时

$$s_j = R(\alpha^j) = q_j(\alpha^j)m_j(\alpha^j) + r_j(\alpha^j) = r_j(\alpha^j)$$

而译码过程的关键在于第二步由 s_j 求 $\sigma(x)$ 的系数 $\sigma_1, \sigma_2, \dots, \sigma_{t-1}, \sigma_t$, 这一步如果通过解线性方程组 $[M][\sigma] = -[S]$, 即:

$$\begin{bmatrix} s_t & s_{t-1} & \cdots & s_1 \\ s_{t+1} & s_t & \cdots & s_2 \\ \vdots & \vdots & \vdots & \vdots \\ s_{2t-1} & s_{2t-2} & \cdots & s_t \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} = - \begin{bmatrix} s_{t+1} \\ s_{t+2} \\ \vdots \\ s_{2t} \end{bmatrix}$$

来求得, 那么其计算量将很大, 计算量和系数矩阵阶数

的三次方成正比。尤其当码长较长, 纠错能力较大时, 计算量也随之加大, 而且要求译码器的运算速度很高。为此, BM 算法的提出解决了求 $\sigma(x)$ 的速度, 从工程上解决了 BCH 码的译码问题。

2.2.1 BM 迭代译码算法基本原理

设

$$\sigma(x) = 1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_t x^t = \prod_{i=1}^t (1 - x_i x) \quad (1)$$

$$S(x) = 1 + s_1 x + s_2 x^2 + \cdots = \sum_{i=0}^{\infty} s_i x^i = \sum_{i=0}^{\infty} \left(\sum_{j=1}^t Y_j x_j^i \right) x^i \quad (2)$$

作乘积

$$S(x)\sigma(x) = \omega(x) = 1 + \omega_1 x + \omega_2 x^2 + \cdots \quad (3)$$

结合(1)式和(2)式将(3)式展开得到

$$\begin{aligned} S(x)\sigma(x) &= 1 + (s_1 + \sigma_1)x + (s_2 + \sigma_1 s_1 + \sigma_2)x^2 \\ &+ \cdots + (s_t + \sigma_1 s_{t-1} + \cdots + \sigma_t)x^t = \\ &1 + \omega_1 x + \omega_2 x^2 + \cdots + \omega_t x^t + \cdots \end{aligned} \quad (4)$$

在(4)式中大于 x^t 的系数均为 0, 因而 ∂_0

$S(x)\sigma(x) \leq 2t$, 由此得出关键方程^[1]:

$$S(s)\sigma(x) \equiv \omega(x) \pmod{x^{2t+1}} \quad (5)$$

当 t 较小时, 应用此关键方程很容易求出 $\sigma(x)$ 的系数, 但当 t 比较大时仍然很麻烦。下面讨论 BM 迭代译码算法。

应用 BM 迭代算法首先要确定合理的初始值, 以此为基础进行迭代, 这里选 $\sigma^{(0)}(x)$ 和 $\omega^{(0)}(x)$, $\sigma^{(-1)}(x)$ 和 $\omega^{(-1)}(x)$, 然后进行第一次迭代, 用 $\sigma^{(0)}(x)$ 和 $\omega^{(0)}(x)$ 表示 $\sigma^{(1)}(x)$ 和 $\omega^{(1)}(x)$, 依次类推, 直到解出满足 $\sigma(x)$ 和 $\omega(x)$ 为止, 其间一共进行 $2t-1$ 次迭代。迭代过程中定义第 $j+1$ 步与第 j 步的差值为 d_j , 则有

$$d_j = s_{j+1} + \sum_{i=1}^{\partial_0 \sigma^{(j)}(x)} s_{j+1-i} \sigma_i^{(j)}$$

式中 $\sigma_i^{(j)}$ 是第 j 次迭代后 $\sigma(x)$ 中 x^i 的系数, $\partial_0 \sigma^{(j)}$ 为 $\sigma^{(j)}(x)$ 的系数, 用 $D(j)$ 表示, 即 $D(j) = \partial_0 \sigma^{(j)}(x)$, 且下面公式成立:

$$\sigma^{(j+1)}(x) = \sigma^{(j)}(x) - d_j d_i^{-1} x^{j-i} \sigma^{(i)}(x)$$

$$\omega^{(j+1)}(x) = \omega^{(j)}(x) - d_j d_i^{-1} x^{j-i} \omega^{(i)}(x)$$

这里 i 是 j 前面的某一行, 并满足 $i - D(i)$ 最大, 且 $d_i \neq 0$, 而

$$D(j+1) = \max(D(j), j - i + D(i))$$

迭代步骤如下:

(1) 初始化

$$\sigma^{(-1)}(x) = 1, \omega^{(-1)}(x) = 0, D(-1) = 0, d_{-1} =$$

1
 $\sigma^{(0)}(x) = 1, \omega^{(0)}(x) = 1, D(0) = 0, d_0 = s_1$
 (2) $j = j + 1$;
 (3) for($j = 1; j \leq 2t; j++$)
 for($i = 1; i \leq \partial_o \sigma^{(j)}; i++$)
 $d_j = s_{j+1} + \sum_{i=1}^{\partial_o \sigma^{(j)}(x)} s_{j+1-i} \sigma_i^{(j)}$
 (4) if($d_j = 0$)
 $\sigma^{(j+1)}(x) = \sigma^{(j)}(x); \omega^{(j+1)}(x) = \omega^{(j)}(x);$
 $(j+1) - D(j+1) = j - D(j);$
 (5) else 计算 d_{j+1} ;
 if($i < j \ \&\& \max(i - D(i)) \&\& d_i! = 0$)
 $\sigma^{(j+1)}(x) = \sigma^{(j)}(x) - d_j d_i^{-1} x^{j-i} \sigma^{(i)}(x);$
 $\omega^{(j+1)}(x) = \omega^{(j)}(x) - d_j d_i^{-1} x^{j-i} \omega^{(i)}(x);$
 (6) if($j == 2t$)
 $\sigma(x) = \sigma^{(j+1)}(x); \omega(x) = \omega^{(j+1)}(x);$
 else 返回(2)
 (7) if($\partial_o \sigma^{(t)}(x) \leq t$) break;
 else 有不可纠的错误
 (8) 经 $2t$ 次迭代退出循环得到所求的差错定位多项式 $\sigma(x)$ 和错误位置多项式 $\omega(x)$ 。

值得注意的是在迭代过程中, 为了使得解具有唯一性, 必须保证 $\sigma(x)$ 的次数最小, 且 $\omega(x)$ 的次数不大于 $\sigma(x)$ 的次数。

2.2.2 二进制 BCH 码 BM 算法的简化

这里先引进在 $GF(2)$ 域中的牛顿恒等式

$$s_1 + \sigma_1 = 0$$

$$s_2 + s_1 \sigma_1 = 0$$

$$s_3 + s_2 \sigma_1 + s_1 \sigma_2 + \sigma_3 = 0$$

⋮

代入关键方程(5) 式得:

$$S(x)\sigma(x) = 1 + \sigma_2 x^2 + \sigma_4 x^4 + \cdots + \sigma_{2t} x^{2t} \equiv \omega(x) \bmod x^{2t+1}$$

由此可见关键方程的奇数次项的系数均为 0, 只有偶次项系数, 所以二进制 BCH 码的 BM 算法有以下特点:

$$1) \omega^{(i)}(x) = \sigma_e^{(i)}(x) \quad i = 0, 1, 2, \cdots, 2t$$

($\sigma_e^{(i)}$ 为 $\sigma^{(i)}(x)$ 的偶部多项式)

$$2) d_{2k+1} = 0 \quad k = 0, 1, 2, \cdots, t-1$$

从以上分析可以给出二进制 BCH 码的 BM 算法步骤:

(1) 初始化 $\sigma^{(-1/2)}(x) = 1, D(-1/2) = 0, d_{-1/2} = 1$
 $\sigma^{(0)}(x) = 1, D(0) = 0, d_0 = s_1$

$$(2) j = j + 1;$$

.....

$$(3) \text{if}(d_j = 0)$$

$$\sigma^{(j+1)}(x) = \sigma^{(j)}(x); D(j+1) = D(j);$$

$$(4) \text{else 计算 } d_{j+1};$$

$$\text{if}(i < j \ \&\& \max(2i - D(i)) \&\& d_i! = 0)$$

$$\sigma^{(j+1)}(x) = \sigma^{(j)}(x) - d_j d_i^{-1} x^{j-i} \sigma^{(i)}(x);$$

$$(5) \text{一直往下迭代, 直到算出 } \sigma^{(t)}(x) \text{ 为止。}$$

可以看出二进制 BCH 码的 BM 算法只需要 t 次迭代, 大大减少了迭代次数^[3]。

2.3 用钱氏 (Chien) 搜索法求 $\sigma(x) = 0$ 的根

求解 $\sigma(x) = 0$ 的根就是确定 $R(x)$ 中哪几位发生了错误, 即确定 α^{n-i} 是不是错误位置数, 那就需要验证 $\alpha^{-(n-i)}$ 是不是 $\sigma(x) = 0$ 的根。若是则第 $n-i$ 位有错, 否则就无错。依次对每一位 $n-i$ ($i = 1, 2, \cdots, n$) 进行检验, 最后得到 $\sigma(x) = 0$ 的根, 然后取反就得到相应的错误位置。这就是钱氏搜索法的基本思想。

3 二进制 (50, 32) 码译码算法 C/C++ 语言的实现

二进制 (50, 32) 码是基于域 $GF(2^6)$ 上的 BCH(63, 45) ($t = 3$) 的缩短码。缩短码有如下性质: 纠错能力与原码相同, 最小汉明距离不变, 生成多项式也不变; 在说明译码原理时, 只需要在接收码字高位处补上 $2^m - 1 - n = 13$ 个 0 即可, 但在实际译码时就不需要这样做, 仍可采用本原长码的译码方法来进行处理。而对于校验矩阵只需要去掉相应的列数 (从左到右) 就行了^[2]。由此 (50, 32) 码的最小汉明距离为 $d_{\min} = 7$, 在实际应用中能满足纠正两个错误的同时检测四个错误 (纠二检四), 其生成多项式为:

$$g(x) = m_1(x)m_3(x)m_5(x) = (x^6 + x + 1)(x^6 + x^4 + x^2 + x + 1)(x^6 + x^5 + x^2 + x + 1) = x^{18} + x^{17} + x^{16} + x^{15} + x^9 + x^7 + x^6 + x^3 + x^2 + x + 1$$

所以校验位数为 $\partial_o g(x) = 18$ 个。

在本算法程序设计中, 涉及有限域多项式乘除法, 因此为了提高计算机纠错的运算速度, 最好的办法就是在程序中先建立一张 $GF(2^6)$ 域上可供查询的各剩余类多项式之间关系的表。 $\alpha \in GF(2^6)$ 域, 为本原多项式 $x^6 + x + 1$ 的根, 以此多项式为模作剩余类, 就能得到 2^6 阶有限域 $GF(2^6)$, 如表 1 所示。

由二进制 BCH 码的编码理论可知, 伴随式只需计算 $s_1 = r_1(\alpha), s_3 = r_3(\alpha^3), s_5 = r_5(\alpha^5)$ 。因为 $s_2 = s_1^2$

$= r_1(\alpha^2), s_4 = s_2^2 = r_1(\alpha^4), s_6 = s_3^2 = r_3(\alpha^6)$, 所以在进行程序设计时只需考虑 3 个伴随式, 这样可以节省计算时间和提高程序运行速度。如果计算出来 $S = 0$, 则说明没有错误发生, 那么程序就退出, 不需要纠错; 如果计算出来 $S \neq 0$, 则说明有错误产生, 程序执行下一步进行纠错。

表 1 以 $x^6 + x + 1$ 为模的有限域 $GF(2^6)$ 的元素

本原域元素	α 多项式	剩余类	对应的多项式
$\alpha^0 = 1$	α^0	1	000001
α	α	x	000010
α^2	α^2	x^2	000100
α^3	α^3	x^3	001000
α^4	α^4	x^4	010000
α^5	α^5	x^5	100000
α^6	$\alpha + 1$	$x + 1$	000011
...
α^{50}	$\alpha^5 + \alpha^4 + \alpha^2$	$x^5 + x^4 + x^2$	110100

因为(50,32)码实际应用在某处理器系统中, 考虑时间、体积、功耗之间的合理性, 所以根据处理器内部的体系结构及流水机制, 最大程度地来降低计算量以减少不必要的等待周期, 所以在算法结构上拟通过三级流水线来完成, 这样错误纠正后能自动回写到存储器和寄存器中。基于上述考虑, 那么计算伴随式为第一级, 差错定位多项式的计算为第二级, 钱氏搜索算法为第三级。在程序设计时就可以分成三个模块, 当然在上述算法结构中, 根据算法程序的需要, 具体模块中还可以细化分成几个功能模块。如涉及到的多项式乘法和除法等就需要相应独立的程序块^[4,5]。这样便于复杂的程序简单化和清晰化。

接下来就是根据前面所述的算法过程进行具体的译码, 最后把各个模块、流水级之间通过调用函数程序来进行编译、调试。其译码算法程序流程图如图 1 所示。

4 小 结

随着计算机技术的发展, 纠错技术也得到越来越广泛的应用。笔者通过对 BCH 码译码算法深入的研究和探讨, 从实际出发选定 BM 算法进行分析, 结合二进制 BCH(50,32)码给出了用 C/C++ 语言实现方法, 并采取流水译码算法结构来提高译码速度和正确性。

参考文献:

[1] 王新梅, 肖国镇. 纠错码——原理与方法[M]. 西安: 西安电子科技大学出版社, 2003.

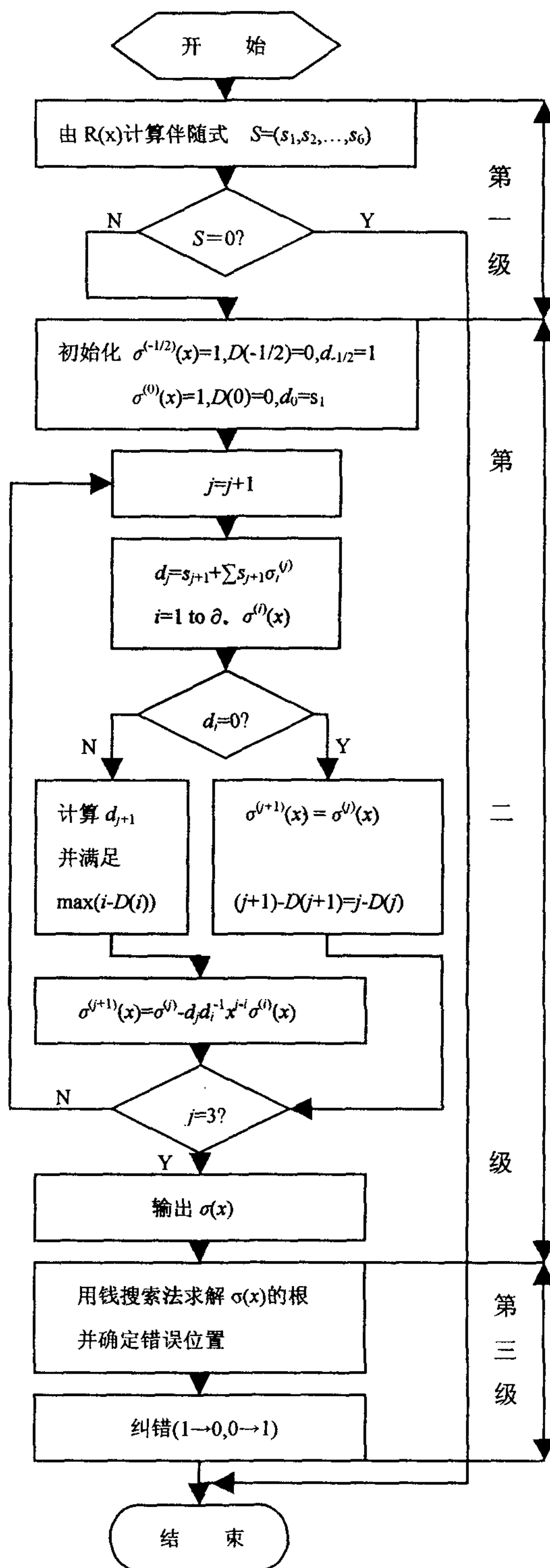


图 1 (50,32)码译码算法程序流程图

[2] 姚明余, 忻鼎稼. BCH 码的一种新的译码方法[J]. 通信学报, 1989(5): 10-14.

[3] Chen C L. High-speed decoding of BCH code[J]. IEEE Trans. on IT, 1981(2): 254-256.

[4] 王新梅. 计算机中的纠错码技术[M]. 北京: 人民邮电出版社, 1999.

[5] 王进祥. RS(255,223)译码器的 VLSI 设计[D]. 哈尔滨: 哈尔滨工业大学, 1999.