[256,252]RS 扩展码的快速译码算法

张玉良[***] 陈晓敏[*]
[*](中国科学院空间科学与应用研究中心 北京 100080)
[**](中国科学院研究生院 北京 100039)

摘 要：该文主要论述[256,252]RS(Reed-Solomon)扩展码的快速译码算法。该算法是通过简单的参数测试来发现接收数据中的错误类型以及错误模式，然后通过得到的错误模式来对接收数据进行错误纠正。与已有的译码算法相比，该算法具有占用硬件资源相对较少，处理时间相对较短的优点，并且在硬件译码器上实现的最高数据处理速率超过400Mbit/s。

关键词：译码算法，RS 扩展码
中图分类号：TN911.22 **文献标识码**：A **文章编号**：1009-5896(2005)12-1947-05

The Fast Decoding Algorithim of [256,252] RS Extended Code

Zhang Yu-liang[***] Chen Xiao-min[*]
[*](*Center for Space Science and Applied Research Chinese Academy of Sciences, Beijing* 100080, *China*)
[**](*Graduate School of the Chinese Academy of Sciences, Beijing* 100039, *China*)

Abstract In this article, the main point is to describe the fast decoding algorithm of [256,252] RS extended code. In order to correct the error in the received data quickly, the algorithm gets the error type and error pattern through simple parameter-comparison, then adds the error pattern to the receive data. Compared to the algorithms in existence, this algorithm has the advantages of using less hardware resources and decoding time. When this algorithm implements in hardware, its throughput is more than 400Mbit/s.

Key words Decoding algorithm, RS extended code

1 引言

从日常生活中的电子设备，如硬盘存储、数字电视等，到大型的通信系统，如深空探测、星际侦察等，人们都可以发现 RS 码的应用踪迹。而且在 RS 码的应用领域不断扩大的同时，对 RS 码的编译码算法所提出的要求也越来越高。本文所论述的快速译码算法正是为了适应工程需要对[256,252]RS 扩展码需要满足的快速硬件译码要求而提出的。虽然参考文献[1]阐述了针对具有纠正两个错误和检测3个错误的 RS 码的快速译码算法，但是对于 RS 扩展码的译码研究来说，该算法却仅有参考价值而不能直接应用。同时，尽管对于 RS 码的译码算法已经有广泛的研究，但是对于 RS 扩展码的译码算法的研究却为数不多的，尤其针对像[256,252]RS 扩展码一样常用于保护固态存储器等器件的 RS 扩展码的硬件译码器的研究更是难以发现。

因此，在借鉴参考文献[1]引入辅助参数的基础上，本译码算法进一步提出了如下的算法思想：在[256,252]RS 扩展码的纠错能力有限的情况下，依据在补充校验位上是否发生错误的事实，采用了直接获得错误位置与错误值的方法。在以上算法思想的指导下，针对某工程的有效载荷数管系统的大容量固态存储中的实际需要，根据下文所阐述的译码算法实现了最高数据处理速率超过 400Mbit/s 的[256,252]RS 扩展码的硬件译码器，而且在同类工作中实现了从耗时的软件译码到实时的硬件译码的首次突破。

2 译码算法的具体方案及其性能分析

2.1 译码算法的算法假设与符号说明

由于[256,252]RS 扩展码是一种基于[255,252]RS 码扩展而来的 RS 扩展码，而[255,252]RS 码只不过是一种具有纠正一个错误和检测两个错误的纠错码，所以，尽管经过扩展而得到的[256,252]RS 扩展码可以纠正两个错误的情况，但是它本身对等于或大于 3 个错误以上的情况并没有直接的检测能力。因此，为了获得更好的译码算法和更加清晰的阐述，假设在所检测的接收码子中不存在发生等于或大于 3 个错误的

2004-06-14 收到，2005-03-21 改回

情况。在此假设成立的情况下，从下文的证明可知，本译码算法不会发生错误译码。同时，由假设可以得到如下性质：

性质 1 $V = C + E$，即$v_i = c_i + e_i$(其中i从 0 到 255)；

性质 2 在每一个E中的所有分量最多有两个不同位置上的值同时不为零，即当$i \neq j$，且E中的e_i与e_j都不为零时，则在E中的其余分量必为零。

在论文中所采用的符号具体说明如下$(c_0, c_1, \cdots, c_{254}, c_{255})$为经过编码算法得到的码子，记为$C$(其中$c_1$，$c_2$和$c_3$分别为对应编码信息经过编码算法所获得的一般校验位，c_0为对应编码信息经过编码算法所获得的补充校验位)；$(v_0, v_1, \cdots, v_{254}, v_{255})$为经过信道传输后所接受到的码子，记为$V$；$(e_0, e_1, \cdots, e_{254}, e_{255})$为与V相对应的错误模式，记为$\underline{E}$；当$e_k$= 0(其中$k$从 0 到 255)时，记$E$为$E_1$；当$e_0 \neq 0$，且$e_k$ = 0(其中k从 1 到 255)时，记E为E_2；当$e_i \neq 0$ (其中i属于 1 到 255 中的任意一个)，且e_k= 0(其中k从 0 到 255，且$k \neq i$)时，记E为E_3；当$e_0 \neq 0$ 和$e_i \neq 0$ (其中i属于 1 到 255 中的任意一个)，且e_k = 0(其中k从 1 到 255，且$k \neq i$)时，记E为E_4；当$e_i \neq 0$ 和$e_j \neq 0$ (其中i与j分别属于 1 到 255 且 $i \neq j$)，且e_k = 0(其中k从 0 到 255，且$k \neq i$同时$k \neq j$)时，记E为E_5；

记：校验子向量$\begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix}$为 $\boldsymbol{S}$，校验矩阵:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & \alpha & \cdots & \alpha^{254} \\ 0 & 1 & \alpha^2 & \cdots & \left(\alpha^2\right)^{254} \\ 0 & 1 & \alpha^3 & \cdots & \left(\alpha^3\right)^{254} \end{pmatrix}$$

为 $\boldsymbol{H}$，α 为 $x^8+x^4+x^3+x^2+1=0$ 的本原元。同时，扩展前的[255,252]RS 码的码生成多项式为$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)$。

2.2 译码算法的具体方案

根据译码算法的算法思想，[256,252]RS扩展码能否纠正错误的情况一共可分为 6 种。(1) 接收数据没有发生错误，则错误模式为E_1；(2) 接收数据发生 1 个错误并且错误位置在补充校验位上，则错误模式为E_2；(3) 接收数据发生 1 个错误但错误位置不在补充校验位上，则错误模式为E_3(4) 接收数据发生两个错误，其中 1 个错误在补充校验位上，另 1 个错误在其他位置上，则错误模式为E_4；(5) 接收数据发生两个错误但都不在补充校验位上，则错误模式为E_5；(6) 接收数据发生其他情况的错误。所以对于[256,252]RS扩展码的译码方案可以有别于其它能够纠正更多错误的RS码的译码方案。其具体译码方案如下：

第 1 步 随着接收数据以字节为单位而串行输入时，译码器使用Horner算法获得接收数据的校验子，即 计算 4 个校验子S_i (i = 0,1,2,3)：

$$\begin{cases} T_j^i = T_{j-1}^i \alpha^i + v_{256-j} \\ S_i = T_{256}^i \end{cases}$$

其中 $j = 1, \cdots, 256$；i = 0,1,2,3；$T_0^i = 0$ 。

第 2 步 在接收到从第 1 步输出的结果后，译码算法计算了 3 个辅助参数 β_j (j =0,1,2)，其中

$$\beta_0 = S_1^2 + S_0 S_2, \beta_1 = S_2^2 + S_1 S_3, \beta_2 = S_1 S_2 + S_0 S_3$$

第 3 步 在接收到从第 2 步输出的结果后，译码算法通过第 2 步输出结果的情况来判断接收数据所发生的错误是上述 6 种错误情况的哪一种。即(1) 当S_i(i = 0,1,2,3)全为零时，为第 1 种错误情况；(2) 当S_i(i = 1,2,3)全为零，但S_0不为零时，为第 2 种错误情况；(3) 当 β_j (j = 0,1,2)全为零，但S_2不为零时，为第 3 种错误情况；(4) 当 β_j (i = 0,2)不为零，S_2不为零，但 $\beta_1 = 0$ 时，为第 4 种错误情况；(5) 当 β_j (j = 0,1,2)全不为零时，为第 5 种错误情况；(6) 当其他情况发生时，为第 6 种错误情况。

第 4 步 (1) 如果判断结果为第 4 种错误情况时，则计算与第 4 种错误情况有关的相应参数。即计算 $e_0 = \beta_0 / S_2$ ，$e_i = S_0 + \beta_0 / S_2$ 。(2) 如果判断结果为第 5 种错误情况时，则计算与第 5 种错误情况有关的相应参数。即计算 $\gamma_0 = S_0 \beta_0 / \beta_2$ ， $\gamma_1 = S_1 \beta_0 / \beta_2$ 。

第 5 步 根据相应的判断结果输出相应的错误模式以及译码算法能否对接收数据进行纠正的信息。即根据 6 种不同的判断结果输出不同的错误模式，分别如下：(1) 当为第 1 种错误模式时，输出E_1以及相应的错误情况；(2) 当为第 2 种错误模式时，输出E_2(其中 $e_0 = S_0$)以及相应的错误情况；(3) 当为第 3 种错误模式时，输出E_3 (其中 $e_i = S_0$ ，i满足 $\alpha^i S_3 + S_2 = 0$) 以及相应的错误情况；(4) 当为第 4 种错误模式时，输出E_4 (其中 $e_0 = \beta_0 / S_2$ ， $e_i = S_0 + \beta_0 / S_2$ ，i满足 $\alpha^i S_3 + S_2 = 0$) 以及相应的错误情况；(5) 当为第 5 种错误模式时，输出 E_5 (其中 $e_i = S_0 + \gamma_1 + \gamma_0 \left(\alpha^{-1}\right)^i$ ， i 满足 $\beta_1 \left(\alpha^2\right)^i + \beta_2 \alpha^i + \beta_0 = 0$ ； $e_j = S_0 + \gamma_1 + \gamma_0 \left(\alpha^{-1}\right)^j$ ， j 满足 $\beta_1 \left(\alpha^2\right)^j + \beta_2 \alpha^j + \beta_0 = 0$)以及相应的错误情况；(6) 当为第 6 种错误模式时，输出E_1以及相应的错误情况。

第 6 步 把错误模式与其相对应输入数据进行相应的模和就可以得到相应的正确数据，但如果错误情况为第 6 种情况时，得到的数据为原始接收数据。

2.3 译码算法的性能分析

从上文的论述可知，本文所阐述的算法是针对[256,252]RS 扩展码的硬件译码器而提出的。所以，对于译码算法的性能分析，可以从译码算法所使用的硬件资源和由该算法设计的硬件译码器的数据处理速率来考虑。假设在硬件设计中对所使用的基本运算采用相同的设计方法，那么本译码算法所使用的基本运算的硬件资源如表 1 所示。

表 1　本文所提译码算法的硬件资源使用表

译码 步骤	加法器 (个数)	单变量乘法器 (个数)	双变量乘法器 (个数)	平方器 (个数)	除法器 (个数)
第 1 步	4	3	0	0	0
第 2 步	3	0	4	2	0
第 3 步	0	0	0	0	0
第 4 步	1	0	2	0	1
第 5 步	5	4	0	0	0
第 6 步	1	0	0	0	0
总计	14	7	6	2	1

显然，本译码算法的第 2 步和第 3 步等效于迭代译码算法中的求解关键方程。同时，求解关键方程又是在硬件译码器设计中降低资源和减少延时的瓶颈。而在求解关键方程上，本译码算法与能够纠正两个错误的部分迭代算法之间的比较如表 2 所示。表 2 中，T_{mult}表示双变量乘法器的处理延时，T_{add}表示加法器的处理延时。同时，平方器的处理延时不大于双变量乘法器的处理延时，并且实现平方器所需要的硬件资源不多于实现双变量乘法器所需要的硬件资源。由此可见，在求解关键方程上，本译码算法在硬件资源使用和关键路径延时上取得了满意的折衷。

进一步分析可知，本译码算法的第 4 步和第 5 步的共同作用在于求取错误位置上的错误值，即等效于钱搜索和 Forney 算法所起的共同作用。而本译码算法和常用的钱搜索与 Forney 算法在硬件资源使用上的区别如表 3 所示。

尽管由表 3 可知，本译码算法多使用了 1 个加法器，但是从整个译码器所使用的硬件资源的角度来考虑的话，本译码算法在硬件资源使用上却远少于迭代译码算法所使用的硬件资源。而且在同时输出错误情况的要求下，本译码算法的整体输入输出延时不超过 270 个处理时钟周期，而迭代译码算法却是难以实现的。最后通过验证，根据本算法设计的硬件译码器的最高数据处理速率可超过 400Mbit/s。

3　译码算法的关键证明

对于本文所阐述的译码算法来说，其关键方面在于算法中所引入的辅助参数和根据辅助参数来获得接受码子的错误情况的正确性。因此，本小节采用排除法来证明上述译码算法在由所引入的辅助参数来判断接受码子的错误情况的做法的正确性。由上文的假设可知，当接受码子不发生等于或大于 3 个错误时，接受码子可发生的错误情况为上述的 5 种不同错误模式。因此，在此假设下可得 5 个结论：

(1) $S_i\,(i = 0,1,2,3)$全为零，是接受码子发生第 1 种错误情况的充要条件；(2) $S_i\,(i = 1,2,3)$全为零，但S_0不为零，是接受码子发生第 2 种错误情况的充要条件；(3) $\beta_j\,(j = 0,1,2)$全为零，但S_2不为零，是接受码子发生第 3 种错误情况的充要

表 2　关键方程的硬件资源使用对比表

使用 算法	加法器 (个数)	双变量乘法器 (个数)	平方器 (个数)	所需处理时钟周期 (个数)	关键路径延时
iBM (Blahut)	5	9	0	6	$> 2(T_{\text{mult}} + T_{\text{add}})$
iBM (Berlekamp)	7	13	0	4	$> 2(T_{\text{mult}} + T_{\text{add}})$
RiBM [2]	7	14	0	4	$T_{\text{mult}} + T_{\text{add}}$
Euclidean [3]	10	24	0	4	$>T_{\text{mult}} + T_{\text{add}}$
Euclidean [4] (folded)	5	5	0	24	$>T_{\text{mult}} + T_{\text{add}}$
本文所提的译码算法	3	4	2	2	$\leqslant T_{\text{mult}} + T_{\text{add}}$

表 3　获取错误值的硬件资源使用对比表

使用算法	加法器(个数)	单变量乘法器(个数)	除法器(个数)
钱搜索和 Forney 算法	5	4	1
本文所提的译码算法	6	4	1

条件；(4) $\beta_j\,(i=0,2)$不为零，S_2不为零，但 $\beta_1=0$，是接受码子发生第 4 种错误情况的充要条件；(5) $\beta_j\,(j=0,1,2)$全不为零，是接受码子发生第 5 种错误情况的必要条件。

3.1 结论(1)的证明

必要性 因为 $\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_1^{'}=\begin{pmatrix}0\\0\\0\\0\end{pmatrix}$， 所以，必要性成立。

充分性 (a)设第 2 种错误情况发生，即错误模式为E_2，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_2^{'}=\begin{pmatrix}e_0\\0\\0\\0\end{pmatrix}$$

因为$S_i=0(i=0,1,2,3)$，所以 $e_0=0$ (与假设矛盾)，即假设(a)不成立。

(b) 设第 3 种错误情况发生，即错误模式为E_3，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_3^{'}=\begin{pmatrix}e_i\\\alpha^i e_i\\\alpha^{2i}e_i\\\alpha^{3i}e_i\end{pmatrix}$$

因为$S_i=0(i=0,1,2,3)$，所以 $e_i=0$ (与假设矛盾)，即假设(b)不成立。

(c) 设第 4 种错误情况发生，即错误模式为E_4，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_4^{'}=\begin{pmatrix}e_0+e_i\\\alpha^i e_i\\\alpha^{2i}e_i\\\alpha^{3i}e_i\end{pmatrix}$$

因为$S_i=0(i=0,1,2,3)$，所以 $\alpha^i e_i=0$， $e_0=e_i$。又因为 $\alpha^i\neq0$，因此 $e_0=e_i=0$ (与假设矛盾)，即假设(c)不成立。

(d) 设第 5 种错误情况发生，即错误模式为E_5，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_5^{'}=\begin{pmatrix}e_i+e_j\\\alpha^i e_i+\alpha^j e_j\\\alpha^{2i}e_i+\alpha^{2j}e_j\\\alpha^{3i}e_i+\alpha^{3j}e_j\end{pmatrix}$$

因为$S_i=0(i=0,1,2,3)$，所以 $\alpha^i e_i=\alpha^j e_j$， $e_i=e_j$，故 $\alpha^{i-j}=1$，因此$i=j$ (与假设矛盾)，即假设(d)不成立。由以上证明可知，结论(1)成立。

3.2 结论(2) 的证明

必要性 因为 $\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_2^{'}=\begin{pmatrix}e_0\\0\\0\\0\end{pmatrix}$，所以，必要性成立。

充分性 (a) 设第 3 种错误情况发生，即错误模式为E_3，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_3^{'}=\begin{pmatrix}e_i\\\alpha^i e_i\\\alpha^{2i}e_i\\\alpha^{3i}e_i\end{pmatrix}$$

因为 $S_1=0$， $\alpha^i\neq0$，所以 $e_i=0$ (与假设矛盾)，即假设(a)不成立。

(b) 设第 4 种错误情况发生，即错误模式为E_4，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_4^{'}=\begin{pmatrix}e_0+e_i\\\alpha^i e_i\\\alpha^{2i}e_i\\\alpha^{3i}e_i\end{pmatrix}$$

因为 $S_1=0$， $\alpha^i\neq0$，所以 $e_i=0$ (与假设矛盾)，即假设(b)不成立。

(c) 设第 5 种错误情况发生，即错误模式为E_5，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_5^{'}=\begin{pmatrix}e_i+e_j\\\alpha^i e_i+\alpha^j e_j\\\alpha^{2i}e_i+\alpha^{2j}e_j\\\alpha^{3i}e_i+\alpha^{3j}e_j\end{pmatrix}$$

因为 $S_1=S_2=S_3=0$， $\alpha^i\neq0$，所以 $\alpha^{i-j}=e_j/e_i$，$\alpha^{2(i-j)}=e_j/e_i$，故，$\alpha^{i-j}=1$，因此 $i=j$(与假设矛盾)，即假设(c)不成立。由以上证明可知，结论(2)成立。

3.3 结论(3)的证明

必要性 因为 $\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_3^{'}=\begin{pmatrix}e_i\\\alpha^i e_i\\\alpha^{2i}e_i\\\alpha^{3i}e_i\end{pmatrix}$，$e_i\neq0,\ \alpha^{2i}\neq0$，所以 $S_2\neq0$， $\beta_0=S_1^2+S_0S_2=\left(\alpha^i e_i\right)^2+e_i\cdot\alpha^{2i}e_i=0$， $\beta_1=S_2^2+S_1S_3=\left(\alpha^{2i}e_i\right)^2+\alpha^i e_i\cdot\alpha^{3i}e_i=0$，$\beta_2=S_1S_2+S_0S_3=\alpha^i e_i\cdot\alpha^{2i}e_i+e_i\cdot\alpha^{3i}e_i=0$，因此，必要性成立。

充分性 (a) 设第 4 种错误情况发生，即错误模式为E_4，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_4^{'}=\begin{pmatrix}e_0+e_i\\\alpha^i e_i\\\alpha^{2i}e_i\\\alpha^{3i}e_i\end{pmatrix}$$

因为 $\beta_0=S_1^2+S_0S_2=\alpha^{2i}e_0e_i=0$， $\alpha^i\neq0$，所以 $e_0=0$ 或 $e_i=0$ (与假设矛盾)，即假设(a)不成立。

(b) 设第 5 种错误情况发生，即错误模式为E_5，则

$$\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_5^{'}=\begin{pmatrix}e_i+e_j\\\alpha^i e_i+\alpha^j e_j\\\alpha^{2i}e_i+\alpha^{2j}e_j\\\alpha^{3i}e_i+\alpha^{3j}e_j\end{pmatrix}$$

因为 $\beta_0=S_1^2+S_0S_2=\alpha^{2j}e_ie_j(\alpha^{2(i-j)}+1)=0$， $\alpha^{2j}\neq0$，所以 $e_i=0$ 或 $e_j=0$ 或 $i=j$(与假设矛盾)，即假设(b)不成立。

由以上证明可知，结论(3)成立。

3.4 结论(4)的证明

必要性　因为 $\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_4^{'}=\begin{pmatrix} e_0+e_i \\ \alpha^i e_i \\ \alpha^{2i}e_i \\ \alpha^{3i}e_i \end{pmatrix}$，$e_0\neq 0$，$e_i\neq 0$，$\alpha^i\neq 0$，所以 $S_2\neq 0$，$\beta_0=S_1^2+S_0S_2=\alpha^{2i}e_0e_i\neq 0$，$\beta_1=S_2^2+S_1S_3=\left(\alpha^{2i}e_i\right)^2+\alpha^i e_i\cdot\alpha^{3i}e_i=0$，$\beta_2=S_1S_2+S_0S_3=\alpha^{3i}e_0e_i\neq 0$，因此，必要性成立。

充分性　设第 5 种错误情况发生，即错误模式为$\boldsymbol{E}_5$，则 $\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_5^{'}=\begin{pmatrix} e_i+e_j \\ \alpha^i e_i+\alpha^j e_j \\ \alpha^{2i}e_i+\alpha^{2j}e_j \\ \alpha^{3i}e_i+\alpha^{3j}e_j \end{pmatrix}$。

因为 $\beta_1=S_2^2+S_1S_3=\alpha^{i+3j}e_ie_j(\alpha^{2(i-j)}+1)=0$，$\alpha^{i+3j}\neq 0$，所以 $e_i=0$ 或 $e_j=0$ 或 $i=j$(与假设矛盾)，即假设不成立。

由以上证明可知，结论(4)成立。

3.5 结论(5)的证明

因为 $\boldsymbol{S}=\boldsymbol{H}\cdot\boldsymbol{E}_5^{'}=\begin{pmatrix} e_i+e_j \\ \alpha^i e_i+\alpha^j e_j \\ \alpha^{2i}e_i+\alpha^{2j}e_j \\ \alpha^{3i}e_i+\alpha^{3j}e_j \end{pmatrix}$，$e_i\neq 0$，$e_j\neq 0$，$i\neq j$，所以 $\beta_0=S_1^2+S_0S_2=e_ie_j\alpha^{2j}(\alpha^{2(i-j)}+1)\neq 0$，$\beta_1=S_2^2+S_1S_3=e_ie_i\alpha^{i+3j}(\alpha^{2(i-j)}+1)\neq 0$，$\beta_2=S_1S_2+S_0S_3=e_ie_j\alpha^{3j}(\alpha^{i-j}+1)^3\neq 0$。

由以上证明可知，结论(5)成立。

4　[256,252]RS 扩展码译码算法的验证方案

在进行译码算法的验证时，如果分别对不同的错误情况加以考虑，而且兼顾到 RS 码是循环码和线性码的事实，那么验证的工作就会事半功倍。由于[256,252]RS 扩展码译码算法对错误个数大于或等于 3 的接受数据并不具有纠错或检测的能力(由[256,252]RS 扩展码的固有性质可知)，那么对译码算法进行验证时就不考虑发生错误的第 6 种情况。

由假设所获得的两个性质、RS 码既是线性码又是循环码的事实以及[256,252]RS 扩展码的译码算法可得，用于验证[256,252]RS 扩展码译码算法的错误模式共有 130 种不同向量。它们分别为：

当第 2 种错误情况发生时，$e_0\neq 0$，且$e_k=0$(其中k从 1 到 255)。即，在第 2 种错误情况中仅存在 1 种错误模式，同时该错误模式代表着 255 个不同的错误。

当第 3 种错误情况发生时，$e_i\neq 0$ (其中i属于 1 到 255 中的任意一个，则选$i=1$)，并且$e_k=0$(其中k从 0 到 255，且$k\neq i$)。即，在第 3 种错误情况中仅存在 1 种错误模式，同时该错误模式代表着 255×255 个不同的错误。

当第 4 种错误情况发生时，$e_0\neq 0$ 和$e_i\neq 0$ (其中i属于 1 到 255 中的任意一个，则选$i=1$)，且$e_k=0$(其中k从 1 到 255，且$k\neq i$)，且$e_i\neq e_0$。即，在第 4 种错误情况中仅存在 1 种错误模式，同时该错误模式代表着 255×255^2个不同的错误。

当第 5 种错误情况发生时，$e_i\neq 0$ 和$e_j\neq 0$ (其中i与j属于 1 到 255 中满足 $i\neq j$的任意整数，则选$i=1$，$|j-128.5|<127$ 且j 为不大于 128 的整数)，$e_i\neq e_j$，且$e_k=0$(其中k从 0 到 255，且$k\neq i$同时$k\neq j$)。即，在第 5 种错误情况中仅存在 127 种错误模式，同时该错误模式代表着 $C_{255}^2\times 255^2$个不同的错误。

由假设所获得的性质 1 以及[256,252]RS 扩展码的线性性可知，对第 1 种情况的验证次数为 130 次。换句话说，在译码器能够在可纠错的情况下，不会把正确的码子误认为错误的码子；同时，通过对具有代表性的错误模式验证后，可以保证在发生可纠正的错误模式时能够获得正确的错误向量。

5　结束语

由于本文所阐述的算法思想是基于[256,252]RS 扩展码自身的特性而提出的，所以，在硬件实现上，本算法获得了使用硬件资源少，同时可以在不低于 400Mbit/s 的数据处理速率环境下稳定工作的译码效果。虽然在阐述译码算法的具体方案和关键证明中限定了[256,252]RS 扩展码的域生成多项式和扩展前的[255,252]RS 码的码生成多项式，但是该译码算法却普遍适用于任意的[256,252]RS 扩展码。尽管如此，对于[256,252]RS 扩展码的进一步译码研究仍然进行着。

参 考 文 献

[1] Lee Yuan Xing, Dend R H, Koh Eng Hean. An on-the-fly decoding technique for Reed-Solomon codes. *IEEE Trans. Magnetics*, 1996, 32(5): 3962 – 3964.

[2] Sarwate D V, Shanbhag N R. High-speed architecture for Reed-Solomon decoders [J]. *IEEE Trans. on VLSI Systems*, 2001, 9(5): 641 – 655.

[3] Shao H M, Truong T K, Deutsch L J, Yuen J H, Reed I S. A VLSI design of a pipeline Reed-Solomon decoder [J]. *IEEE Trans. on Comput*., 1985, C-34(5): 393 – 403.

[4] Berlekamp E R, Seroussi G, Tong P. Reed-Solomon Codes and Their Applications. Wicker S B and Bhargava V K, Eds. NJ: IEEE Press, 1994，第十章.

张玉良：　男，1980 年生，硕士生，研究信道编译码技术.

陈晓敏：　女，1951 年生，研究员，主要研究方向包括空间综合电子技术、卫星计算机系统和信号与信息处理.