Website：ycyk.brit.com.cn

基于高速存储平台的高性能RS编译码器设计

丁 琳[1]，孙建伟[2]，武振平[2]
（1 北京宇航系统工程研究所 北京 100076；
2 北京遥测技术研究所 北京 100076）

摘要：目前，星载高速存储设备中采用商用RS编译码IP核来实现数据纠错功能，能够实现的编译码最高速率为800 Mbps，只能依靠多个IP核同时工作达到吉比特高速数据存取速率的要求。星载存储数据发生错误的主要原因是存储区单粒子翻转和存储介质本身特性产生的单比特数据错误。针对星载存储数据的误码特性，本文提出一种RS编译码改进算法，通过对编码算法中的剩余多项式及译码算法中的伴随多项式进行降次处理，减小编译码过程中运算的迭代次数及计算量，以及对编译码算法中的基本运算单元有限域乘法器采用子项复用技术，实现对传统RS编译码算法的改进。结果表明改进后的编译码器能达到最高数据速率为10.5 Gbps，编码器资源较单个商用IP核减少15%，译码器资源减少40%，能够满足后续高速存储平台的应用要求。

关键词：高速存储平台；RS编译码；数据纠错
中图分类号：V44 **文献标志码：**A **文章编号：**2095-1000(2024)03-0058-07
DOI：10.12347/j.ycyk.20231221003
引用格式：丁琳，孙建伟，武振平．基于高速存储平台的高性能RS编译码器设计[J]．遥测遥控，2024，45(3)：58–64.

Design of Highly Efficient RS Encoder and Decoder Based on High-speed Memory Platform

DING Lin[1]，SUN Jianwei[2]，WU Zhenping[2]
（1. Beijing Institute of Astronautical Systems Engineering, Beijing 100076, China;
2. Beijing Research Institute of Telemetry, Beijing 100076, China）

Abstract: Commercial IP core is currently used in the development of high-speed memory device to achieve date error correction,the maximum achievable encoding and decoding speed is 800Mbps，which can only rely on multiple IP cores working simultaneously to meet the requirements of gigabit high-speed data access rate. In view of the bit error characteristics of spaceborne storage data, this paper proposes an improved RS decoding algorithm, which reduces the number of iterations and the amount of computation in the decoding process by downgrading the residual polynomial in the encoding algorithm and the syndrome polynomial in the decoding algorithm, and adopts the sub-term multiplexing technology for the finite field multiplier of the basic operation unit in the decoding algorithm. The implementation results indicate that the maximum speed the encoder and decoder can achieve 10.5 Gbps，while the encoder resources is reduced by 15% and decoder resources reduced by 40% compared to a single commercial IP core, which can meet the application needs of high-speed of memory platform.

Keywords: High-speed of memory platform; RS encoder and decoder; Data error correction
Citation: DING Lin, SUN Jianwei, WU Zhenping. Design of Highly Efficient RS Encoder and Decoder Based on High-speed Memory Platform[J]. Journal of Telemetry, Tracking and Command, 2024, 45(3): 58–64.

0 引言

由于空间单粒子效应和存储介质本身的特性影响，星载数据存储设备中存储的数据会发生一定概率的错误，因此需要采用纠错编译码技术提高星载数据存储的可靠性。目前常采用的纠错编译码技术主要有：汉明码、RS(Reed-Solomon，里得-所罗门)码及LDPC(Low Density Parity Check，低密度奇偶校验)码。汉明码只能纠正一位错误，检测出两位错误、随着存储芯片页容量增大，汉

收稿日期：2023-12-21；修回日期：2024-03-01

明码已不能满足当前多数NAND Flash存储芯片的纠错要求[1]。LDPC编译码算法复杂、硬件实现成本较高，在NAND Flash中的应用还比较少[2]。RS码纠正突发差错和随机差错的能力很强，因而目前广泛应用于数据存储系统的差错控制方案中。

国内外对RS编译码进行了大量的算法研究。国外Peterson、Gore和Zierler提出了PGZ算法，首次解决了RS码的译码问题。Berlekamp提出了迭代译码算法，在工程上解决了RS码的译码问题，易于计算机实现。Massey对Berlekamp迭代译码算法进行改进得到的Berlekamp-Massey(BM)算法，并提出了更适合VLSI(超大规模集成电路)电路设计的算法架构，由此RS码开始在实际应用中得到普及[3]。Sugiyama和Kasahara等发现Euclid算法可以用于RS码译码[4]，Euclid算法与BM算法的主要差别在于迭代过程，Euclid迭代是基于多项式分解求两多项式最大公因式的过程，结构比较规整[5]。Sarwate和Shanbhag根据BM算法的改进算法iBM算法提出riBM算法，成为公认性能最高的RS译码算法[6]。国内周训、何旭等在riBM算法基础上对硬件实现进行了部分改进，提出了TiBM算法，增加了关键方程求解各模块之间的资源共享，使译码性能进一步提高。

现有星载存储设备采用RS编译码商用IP核来实现数据纠错的功能，单个RS编码或译码IP核的最高速率为800 Mbps。随着卫星载荷的发展，星载高速存储设备需要满足的数据存取速率越来越高，在研型号单个存储模块能够达到的最大数据存取速率为10 Gbps，为了满足指标要求只能采用多个IP核并行的方式来实现数据纠错。在星上硬件资源紧张的情况下，现有RS编译码方式已经不能很好地满足任务需求。星载存储数据发生的主要原因为单粒子翻转和存储介质本身特性产生的单比特数据错误，本文针对存储数据的误码特性对传统RS编译码算法进行改进，能够实现硬件资源占用较少并且在一个时钟周期内完成单比特误码的纠错运算，有效提升RS编译码的性能。

1　实现目标

通过对国内外RS编译码算法研究现状分析，目前RS编译码器均为串行编译码算法，RS编译码IP核也采用串行编译码算法，能实现的编译码速率较低，不能适应高速的存储速率，并且现有的RS编译码IP核占用大量的硬件资源，对空间环境的适应性较差。RS编译码IP核为商用IP核，实现代码用户不可见，不利于后续的ASIC设计。

本文的研究目标即对RS编译码算法进行改进使其适用于目前的高速存储平台，用该算法实现的RS编译码器相对于现有IP核能节约大量的硬件资源，在一个时钟周期内就能够完成编译码的运算，使得编译码速度能满足10 Gbps的读写速率要求。

2　传统RS编译码算法

RS码是纠错能力最强、编码效率最高的线性分组码，是一种多进制的BCH码[7]。RS(n,k)码可以由m、n和k三个参数表示，m表示码元取自伽罗华域GF(2^m)，一个码元包含m个bit。n表示码字长度，k表示信息长度。可纠正t个错误的RS码有如下参数：

码长：$n=2^m-1$；

监督符号数目：$n-k=2t$；

最小距离：$d_{\min}=2t+1$；

RS码的生成多项式的一般形式为

$$G(x)=\prod_{i=0}^{n-k-1}(x-\alpha^{K_0+i}) \tag{1}$$

其中，K_0是偏移量，通常取0或1。$K_0=0$时，生成多项式如下:

$$G(x)=(x-1)(x-\alpha)\ldots(x-\alpha^{2t-1})= g_0+g_1x+g_2x^2+\ldots+g_{2t-1}x^{2t-1}+x^{2t} \tag{2}$$

2.1　传统RS编码算法

无论是待编码的信息多项式为

$$M(x)=m_0+m_1x+m_2x^2+\ldots+m_{k-1}x^{k-1} \tag{3}$$

其中，$k=n-2t$。

码元多项式为

$$C(x)=c_0+c_1x+c_2x^2+\ldots+c_{n-1}x^{n-1} \tag{4}$$

剩余多项式为

$$R(x)=Q_0+Q_1x+\ldots+Q_{n-k-1}x^{n-k-1} \tag{5}$$

因

$$\frac{M(x)x^{n-k}}{G(x)}=Q(x)+\frac{R(x)}{G(x)} \tag{6}$$

两边同乘以$G(x)$，得到

$$M(x)x^{n-k}=Q(x)G(x)+R(x) \tag{7}$$

在硬件实现中，RS编码电路由线性反馈移位寄存器实现，如图1所示。

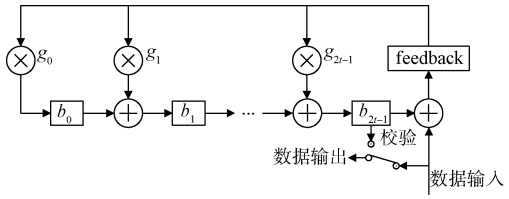

图 1　RS 编码器实现电路

Fig.1　RS encoder implementation circuit

图 1 电路实现的编码步骤如下：将所有寄存器值清零，k 个信息码元依次输入线性反馈移位寄存器电路中进行运算，同时依次输出；k 个时钟周期后，开关切换至校验码元输出，$2t$ 个校验位依次从 b_0~b_{2t-1} 寄存器中输出；n 个时钟周期后，所有码元串行输出结束[8]。

2.2　传统 RS 译码算法

RS 译码过程为计算伴随式、求错误位置多项式、求错误位置数、求解错误值和译码输出。RS 译码器的实现框图见图 2[9,10]。

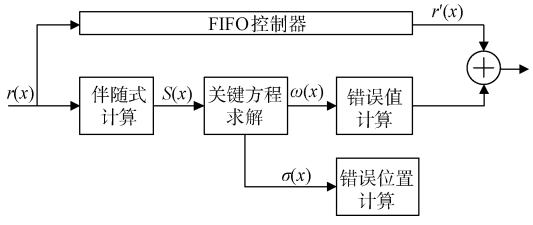

图 2　RS 译码器实现框图

Fig.2　Implementation block diagram of RS decoder

① 伴随式求解

伴随式系数的计算公式为

$$S_i = r(\alpha^i) = \left(\ldots(r_{n-1}\alpha^i + r_{n-2})\,\alpha^i + \ldots r_1\right)\alpha^i + r_0 \quad (8)$$

其中，$0 \leqslant i \leqslant 2t-1$，$r_{n-j}$ 为接收到的第 j 个码元；α^i 为伽罗华域第 $i+1$ 个元素。

② 关键方程求解

关键方程如下

$$\omega(x) = [S(x)\sigma(x)] \bmod x^{2t} \quad (9)$$

关键方程求解目前采用 Berlekamp 算法或 Euclid 算法来实现。硬件上常用 Berlekamp 算法来实现，因其采用规整的脉动阵列，硬件实现更为简单。通过求解关键方程得到错误位置多项式。

③ 错误纠正

$\sigma(x)$ 和 $\omega(x)$ 确定下来后，可用 Chien 搜索算法 $\sigma(x)=0$ 找出错误位置

$$x = \beta_i^{-1} \quad (10)$$

并通过 Forney 算法计算出错误值[11]

$$e_{ji} = \frac{\omega(\beta_i^{-1})}{\sigma_{\text{odd}}(\beta_i^{-1})} \quad (11)$$

$$\sigma_{\text{odd}}(x) = \sigma_1 x + \sigma_3 x^3 + \ldots + \sigma_{\text{todd}} x^{\text{todd}} \quad (12)$$

3　改进算法

由上述传统算法可知，实现编译码的过程需要进行大量的迭代运算，要提高编译码效率可通过以下方式：① 单个 RS 编译码器仅能处理串行输入的码流，若要提高数据处理带宽只能使用多个编译码器并行计算，以牺牲硬件资源为代价；② 要提高译码运算效率，需要提高内部计算模块的时钟频率，时钟频率过高有可能会引起时序问题；③ 需要对传统译码算法进行改进，采用流水并行处理方式缩短计算时间，但译码 IP 核实现代码不透明，需要重新设计算法，增加开发成本。传统的 RS 编译码算法可以纠正一个码组中任意 t 个码元错误，而基于星载存储数据的单比特误码特性[12]，可以将算法改进为仅纠正 1 个码元错误，改进算法能够极大减小编译码过程中运算的迭代次数及计算量。

基于现有高速存储硬件平台，本文选择 RS(24, 22) 码型对改进算法进行设计，其他码型可在 RS(24, 22) 码型基础上对算法进行适应性更改，以适应硬件平台的变化。

3.1　改进 RS 编码算法

采用的 RS(24, 22) 码定义在 GF(2^8) 域上，一个码元为 8 bit 宽度。信息码 $k=22$，码长 $n=24$，能够纠正的最大随机错误为 $t=(n-k)/2=1$ 个字节。GF(2^8) 的本原多项式为

$$P(x) = x^8 + x^4 + x^3 + x^2 + 1 \quad (13)$$

其中，α 定义为 $P(x)=0$ 的根，是 GF(2^8) 的本原元，由式(2)可知，生成多项式为

$$G(x) = (x-1)(x-\alpha) \quad (14)$$

设 22 个信息码为 m_{21}，m_{20}，…，m_0，则信息码多项式为

$$M(x) = m_0 + m_1 x + m_2 x^2 + \ldots + m_{21} x^{21} \quad (15)$$

由于模二加与模二减相同，并且 $n-k=2$，则根据式(7)可得到

$$M(x)\,x^2 + R(x) = Q(x)\,G(x) \quad (16)$$

分别代入 $G(x)=0$ 的根 $x=1$ 和 $x=\alpha$，令等式右侧 $Q(x)G(x)=0$，则上式变为由 Q_1 和 Q_0 共 2 个未知数、2 个方程组成的方程组。解该方程组，求

得的Q_1和Q_0即所求校验码，因此仅需要一个时钟周期即可完成编码计算。

3.2　改进 RS 译码算法

根据传统RS译码算法，译码的第一步即通过接收多项式$r(x)$计算出$n-k$个伴随式的值S_0，S_1，…，S_{2t-1}，把查找错误码字的范围从2^n缩减到2^{n-k}。伴随式的值仅与校验码有关，若伴随式值全为0则说明接收到的码元没有出现错误；若伴随式值不全为0，则说明接收到的码字中存在错误，需进一步查找错误位置并对错误值进行纠正。

改进RS码译码可总结为以下4个步骤：

① 计算伴随多项式的系数S_0和S_1；

② 计算错误位置；

③ 计算错误值；

④ 与接收码元相加得到正确的码元序列。

假设输入译码器的码字多项式为

$$c(x)=c_0+c_1x+c_2x^2+\ldots+c_{23}x^{23} \tag{17}$$

错误多项式为

$$e(x)=e_0+e_1x+e_2x^2+\ldots+e_{23}x^{23} \tag{18}$$

则接收多项式为

$$r(x)=r_0+r_1x+r_2x^2+\ldots+r_{23}x^{23}=c(x)+e(x) \tag{19}$$

其中，$r_i=c_i+e_i$，$i=0,1,\cdots 23$。

已知错误值仅有1个e_v，错误位置在x_v，根据式(8)和式(19)有

$$S_i=r(\alpha^i)=c(\alpha^i)+\mathrm{e}(\alpha^i) \tag{20}$$

因为$c(\alpha^i)=0$，则

$$S_i=e(\alpha^i) \tag{21}$$

经简化后的伴随多项式只有两个系数S_0和S_1，根据式(8)，S_0和S_1计算如下

$$\begin{aligned}S_0&=m_{21}+m_{20}+m_{19}+\ldots+m_0+Q_1+Q_0\\ S_1&=m_{21}\alpha^{23}+\ldots+m_1\alpha^3+m_0\alpha^2+Q_1\alpha+Q_0\end{aligned} \tag{22}$$

如果信息码和校验码没有错误，S_0和S_1应该为0，若不为0，说明发生了错误[13]。若发生了一个码元错误，则有

$$\begin{aligned}S_0&=e_v\\ S_1&=e_vx_v=S_0x_v\end{aligned} \tag{23}$$

其中，$x_v\in\{\alpha^0,\alpha^1,\ldots\alpha^{23}\}$。

根据式(23)仅需要一个时钟周期则能求出错误位置和相应的错误值，完成译码计算。

3.3　有限域乘法器的实现和优化

本文主要研究$m=8$时本原多项式$P(x)=x^8+x^4+x^3+x^2+1$的情况，使用有限域上空间复杂度低、时间延迟少的Mastrovito乘法器构造乘数$\boldsymbol{M}$矩阵，得到$\boldsymbol{M}$矩阵为如下形式

$$\boldsymbol{M}=\begin{bmatrix}
a_0 & a_7 & a_6 & a_5 & a_4 & a_3+a_7 & a_2+a_6+a_7 & a_1+a_5+a_6+a_7\\
a_1 & a_0 & a_7 & a_6 & a_5 & a_4 & a_3+a_7 & a_2+a_6+a_7\\
a_2 & a_1+a_7 & a_0+a_6 & a_5+a_7 & a_4+a_6 & a_3+a_5+a_7 & a_2+a_4+a_6+a_7 & a_1+a_3+a_5+a_6\\
a_3 & a_2+a_7 & a_1+a_6+a_7 & a_0+a_5+a_6 & a_4+a_5+a_7 & a_3+a_4+a_6+a_7 & a_2+a_3+a_5+a_6 & a_1+a_2+a_4+a_5\\
a_4 & a_3+a_7 & a_2+a_6+a_7 & a_1+a_5+a_6+a_7 & a_0+a_4+a_5+a_6 & a_3+a_4+a_5 & a_2+a_3+a_4 & a_1+a_2+a_3+a_7\\
a_5 & a_4 & a_3+a_7 & a_2+a_6+a_7 & a_1+a_5+a_6+a_7 & a_0+a_4+a_5+a_6 & a_3+a_4+a_5 & a_2+a_3+a_4\\
a_6 & a_5 & a_4 & a_3+a_7 & a_2+a_6+a_7 & a_1+a_5+a_6+a_7 & a_0+a_4+a_5+a_6 & a_3+a_4+a_5\\
a_7 & a_6 & a_5 & a_4 & a_3+a_7 & a_2+a_6+a_7 & a_1+a_5+a_6+a_7 & a_0+a_4+a_5+a_6
\end{bmatrix} \tag{24}$$

当RS码型确定时，有限域乘法器中乘数是固定的，此时有限域乘法器可以由一组异或门来实现，有效减少了逻辑资源的占用率。

固定常数乘法器通过将乘数a的分量值a_1~a_7代入上式可得。式中存在大量的公共冗余项，可以利用这些公共冗余项对乘法器进行优化从而减少硬件资源的占用率。

通过子项复用技术来减少常数乘法器的实现规模，使用新的逻辑变量代替这些重复出现的逻辑项就可以减少不必要的冗余。这一方法可以用贪婪算法实现。其基本思想是首先搜索列空间中能产生最多的公共逻辑组合项的两列，将它们按位与后归并成新的一列添加至$\boldsymbol{M}$矩阵的末尾。在包含新增列的扩大列空间中重复上述过程，直至任何两列按位与之后的汉明权重不超过1。在计算过程中，尽量将公共项出现次数多的项提取出来，以利于进行二次优化。表1给出了GF(2^8)域在生成多项式$P(x)=x^8+x^4+x^3+x^2+1$下的前17个常数乘法器的优化结果。由表1可以看出，使用优化后的常数乘法器可以进一步节约硬件资源。

4　实现结果与分析

改进RS编译码算法可以在一个时钟周期内完成编译码的运算，纠正一个字节的数据错误，从

表 1　GF(2^8)上 $x^8+x^4+x^3+x^2+1$ 常数乘法器的改进结果

Table 1　Improvement results of constant multiplier based on $x^8+x^4+x^3+x^2+1$ of GF(2^8)

幂次	改进前异或门数量	改进后异或门数量	优化率(%)
0	/	/	/
1	3	3	0
2	6	5	16.7
3	9	8	11.1
4	12	10	16.7
5	16	12	25
6	19	14	26.3
7	21	16	23.8
8	23	18	21.7
9	22	17	22.7
10	21	16	23.8
11	21	16	23.8
12	21	15	28.6
13	20	16	20
14	21	14	33.3
15	23	15	34.8
16	25	12	52

而满足 10 Gbps 的读写速度。

以某型号高速存储平台作为算法的植入平台，选用的硬件平台存储介质为 IO 8 bit 宽度 FLASH 芯片，24 个 FLASH 芯片并行存储，因此选用 RS 码型为 RS(24, 22)。并行数据宽度为 192 bit，其中包括 176 bit 的有效数据和 16 bit 的校验数据。改进 RS 编译码在系统中的模块示意图如图 3 所示[14]。

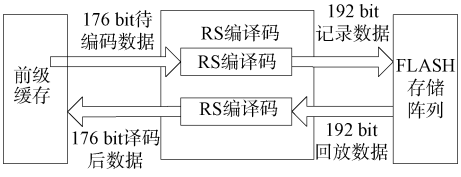

图 3　改进 RS 编译码模块示意图

Fig.3　Schematic diagram of improved RS encoder and decoder module

工作时钟为 60 MHz，图 1 实现的 RS 编译码速度为 $22\times8\times60=10.5$ Gbps，能满足 10 Gbps 的读写需求，而单个 RS 编译码 IP 核实现的编译码的速率仅为 $8\times60\times(22/24)=440$ Mbps，远不能满足 10 Gbps 的读写需求，只能通过多个 RS 编译码 IP 核并行来实现，如图 4 所示。

通过故障注入的方式验证改进 RS 编译码算法

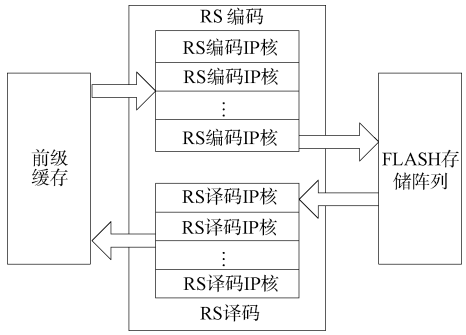

图 4　RS 编译码 IP 核实现示意图

Fig.4　Implementation diagram of RS IP core for encoder and decoder

的功能正确性，将 192 bit 编码数据随机注入 1 bit 错误，验证图如图 5 所示。

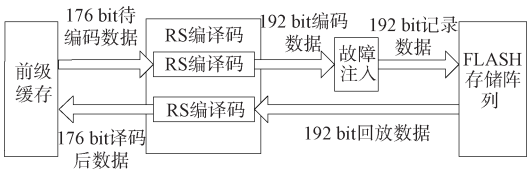

图 5　RS 编译码故障注入功能验证框图

Fig.5　Verification block diagram of fault injection for RS encoder and decoder

故障注入验证改进 RS 编译码功能正确后，将 RS 编译码模块注入星载存储板载 FPGA 逻辑中验证改进 RS 编译码模块代码的适应性。存储 FPGA 实现的功能框图如图 6 所示。其中虚线框中的编译码功能由多个 IP 核并行实现，使用本文研究的改进 RS 编译码器对其进行替换以验证功能及时序的正确性。

采用改进编码及译码模块对原编译码 IP 核模块进行等位替换后进行板级验证，试验结果表明，由于存储介质本身特性产生的单比特误码均被纠正，改进的编译码模块工作稳定，实现了与原有商用编译码 IP 核相同的纠错性能。

使用本文改进算法和商用 IP 核[15,16]实现的 RS 编译码器硬件性能对比见表 2。

对比结果如下：

① 改进后的 RS 编码器和译码器均能实现大于 10 Gbps 的编译码速率，可以满足当前高速存储对数据纠错的需求；

② 改进后的 RS 编码器相对于单个商用 IP 核布局布线后的硬件资源减少 15%；

③ 改进后的 RS 译码器相对于单个商用 IP 核布

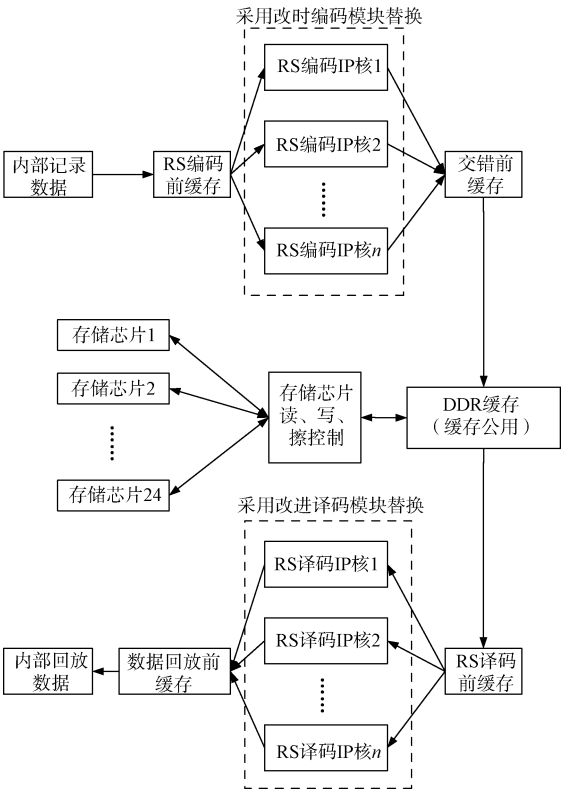

图 6　存储 FPGA 实现功能框图

Fig.6　Block diagram of storage FPGA implementation

表 2　本文和商用 IP 核实现的 RS 编译码器硬件性能对比

Table 2　Comparison of performance between this paper and commercial IP core

实现方法	单个 RS 编码 IP 核	本文设计编码器	单个 RS 译码 IP 核	本文设计译码器
资源（slice）	47	40	836	500
最高频率	733 Mbps（100 MHz）	10.5 Gbps（60 MHz）	733 Mbps（100 MHz）	10.5 Gbps（60 MHz）

局布线后的硬件资源减少 40%。

5　结束语

本文的研究成果后续可以应用于高速数据处理领域，实现的改进 RS 编译码器能满足在研星载存储设备最高的读写速率要求。相对于目前使用的 RS 编译码商用 IP 核在硬件性能上有大幅度的提高，并且实现代码自主透明，具有通用性和灵活性，易于星载设备 FPGA 实现，能够满足星载存储设备高速率、高可靠性的数据存储要求。

参考文献

[1] 吴智龙 . 基于 NAND Flash 的差错控制算法研究[D]. 广州: 广东工业大学, 2014.
WU Zhilong. The research of error control algorithm based on NAND Flash[D]. Guangzhou: Guangdong University of Technology, 2014.

[2] 彭福来, 于治楼, 陈乃阔, 等 . 面向 NAND Flash 存储的纠错编码技术概述[J]. 计算机与现代化, 2017, 11: 35-40.
PENG Fulai, YU Zhilou, CHEN Naikuo, et al. Overview on error correction code technologies for NAND Flash memory[J]. Computer and Modernization, 2017, 11: 35-40.

[3] 梁志斌 . 应用于 RS 译码器的新型高效 Berlekamp-Massey 算法[D]. 天津: 天津大学, 2014.
LIANG Zhibin. Efficient Berlekamp-Massey algorithm and architecture for Reed-Solomon decoder[D]. Tianjin: Tianjin University, 2014.

[4] 周训 . 基于 BM 算法的 RS 译码器 IP 核设计[D]. 成都:电子科技大学, 2009.

[5] LEE H. An ultra high-speed Reed-Solomon decoder[A]. IEEE International Symposium on circuits and systems, 2005(2): 1036-1039.

[6] SARWATE D V, SHANBHAG N R. High-speed architectures for Reed-Solomon decoders[J]. IEEE Transactions Very Large Scaleintegration(VLSI) Systems, 2001, 9(5): 641-655.

[7] 赵琦, 刘荣科 . 编码理论[M]. 北京: 北京航空航天大学出版社, 2009: 132-154.

[8] 向良军, 王梓斌, 金国平, 等 . 高速 RS 编译码器的设计及其 FPGA 实现[J]. 计算机工程与应用, 2012, 48(1): 64-67.
XIANG Liangjun, WANG Zibin, JIN Guoping, et al. Design and FPGA implementation for high-speed RS coding and decoding[J]. Computer Engineering and Applications, 2012, 48(1): 64-67.

[9] 张绍练, 高世杰, 吴志勇 . RS 码仿真与基于 RiBM 算法的硬件实现[J]. 中国光学, 2013, 6(2): 171-178.
ZHANG Shaolian, GAO Shijie, WU Zhiyong. Simulation of RS codes and hardware implementation based on RiBM algorithm[J]. Chinese Optics, 2013, 6(2): 171-178.

[10] 朱悦丰 . RS 译码算法的研究和 FPGA 设计[D]. 南京: 东南大学, 2015.
ZHU Yuefeng. Research and FPGA design of Reed-Solomon decoding algorithm[D]. Nanjing: Southeast

University, 2015.

[11] FORNEY G. On decoding BCH codes[J]. IEEE Transactions on Information Theory, 1965, 11(4): 549-557.

[12] 张文静, 姚智慧 . NAND Flash控制器中RS码的设计与验证[J]. 计算机工程与设计, 2013, 34(7): 2590-2594.
ZHANG Wenjing, YAO Zhihui. Design and verification of RS based NAND flash controller[J]. Computer Engineering and Design, 2013, 34(7): 2590-2594.

[13] 张宇宁, 杨根庆, 李华旺, 等 . 星载高速海量存储系统的并行RS纠错方法[J]. 航天控制, 2009, 27(3): 86-89.
ZHANG Yuning, YANG Genqing, LI Huawang, et al. Parallel Reed-Solomon error correction for spaceborne mass memory system[J]. Aerospace Control, 2009, 27(3): 86-89.

[14] 贾春亮, 周建华, 胡剑平 . 一种新型多模式高速存储器[J]. 遥测遥控, 2013, 34(2): 68-71.
JIA Chunliang, ZHOU Jianhua, HU Jianping. A novel multi-mode high-speed memory[J].Journal of Telemetry, Tracking and Command, 2013, 34(2): 68-71.

[15] Xilinx. LogiCORE IP Reed-Solomon decoder v8.0 data Sheet(DS862)[DS]. Xilinx, 2011: 1-32.

[16] Xilinx. LogiCORE IP Reed-Solomon encoder v8.0 product guide(PG025)[DS]. Xilinx, 2012: 1-30.

[作者简介]

丁 琳 1986年生，硕士，工程师。

孙建伟 1985年生，本科，工程师。

武振平 1986年生，硕士，工程师。

(本文编辑：杨秀丽)