高技术通讯 2010 年 第 20 卷 第 12 期:1274 ~ 1280

doi:10.3772/j.issn.1002-0470.2010.12.011

基于移位多项式基优化并行 RS 伴随式计算电路的方法①

张 亮② 王志功③ 胡庆生

(东南大学射频与光电集成电路研究所 南京 210096)

摘 要 研究了 RS 译码器的并行伴随式计算电路的结构优化,分别推导了并行度能整除和不能整除码长时的并行伴随式计算的表达式,并设计了相应的电路。针对并行实现会增加电路复杂度的问题,通过适当的变换,采用移位多项式基的方法,设计了低复杂度的并行伴随式计算改进电路。改进结构不仅降低了电路中有限域加法器的复杂度,并且通过将原有的多个小规模有限域乘法器简化为一个较大规模的乘法器,使得乘法器的复杂度也在很大程度上得到了降低。对并行度为 8 的 RS(2040,2024)和 RS(255,239)译码器的实验研究表明,上述的结构实现方法可比迭代匹配算法(IMA)节省约 30% 的资源,当并行度为 64 时,资源节省可达到 50%。

关键词 里德-所罗门(RS)译码器,并行伴随式计算电路,移位多项式基,低复杂度结构

0 引 言

里德-所罗门(Reed-Solomon, RS)译码器已广泛应用于各种通信系统的前向纠错中,而且对 RS 编、译码器的研究也相对成熟。但 RS 译码器中的伴随式计算电路因计算时间较长,为适应通信系统容量不断提高的要求,需采用并行结构,而并行结构的设计却带来了复杂度的增大,因而降低复杂度则成为伴随式计算电路并行设计的重点。基于这种情况,本文研究了 RS 译码器的并行伴随式计算电路的结构优化,提出了基于移位多项式基的并行 RS 伴随式计算电路优化方法,并对相应的电路实现复杂度进行了分析。

1 研究背景

RS 译码器通常包括 4 个模块:伴随式计算电路、解关键方程电路、钱氏搜索电路和错误值计算电路。伴随式计算电路根据接收的码字来计算出伴随式,解关键方程电路[1,2]则根据伴随式计算的结果获得错误位置多项式和错误值多项式,钱氏搜索电路[3]则依次把所有错误位置代入错误位置多项式并根据错误位置多项式的值确定该位置是否有错,最后,错误值计算电路采用福尼算法完成纠错功能。

随着光通信系统容量的不断提升,对 RS 译码器的速度也提出了越来越高的要求。为了满足吞吐量的要求,往往需要采用并行的 RS 译码器结构。尤其是其中的伴随式计算电路和钱氏搜索电路,由于需要的处理时间最长,通常采用并行结构。例如,早在 2002 年,Song[4]就提出了并行度为 3 的并行伴随式计算电路,并用该电路实现了单通道吞吐率为 2.5G/s 的 RS 译码器;Lee[5]也采用并行度为 2 的并行伴随式计算电路实现了单通道吞吐量为 10Gb/s 的 RS 译码器。但这些电路结构都局限于特定码型的实现,不能适用于其他码型。同时,并行结构是以增加复杂度为代价来提高速度的,因此,降低复杂度是并行设计中需要考虑的重要问题。

近年来,在并行结构优化方面,陆续出现了不少有影响的研究结果。例如,针对伴随式计算主要是有限域乘法和加法运算的特点,文献[6]提出了迭代匹配算法(iterative matching algorithm, IMA),该算法可以针对单个系数矩阵或多个系数矩阵进行,采用贪婪算法寻找可以合并的子表达式,进而利用共享子表达式的方法减少乘法器中异或门的数量。文献[7]则提出了全局优化算法,用来寻找全局最优的重复子表达式以进一步减少乘法器的复杂度。然而,这些算法均只适用于乘法器,不适用于加法器。

① 863 计划(2006AA01Z284)资助项目。
② 男,1982 年生,博士;研究方向:集成电路设计;E-mail:l.zhang.sd@163.com
③ 通讯作者,E-mail: zgwang@seu.edu.cn
(收稿日期:2009-08-26)

本文重点研究了并行伴随式电路的结构优化。与已有方法不同的是,本文通过对伴随式表达式的适当变换,将其表示成移位多项式基的形式,不仅减少了乘法运算的复杂度,还大大减少了加法运算的复杂度,而且并行度越高,优化效果越明显。本文推导了码长能整除和不能整除并行度时的伴随式表达式,给出了相应的电路结构,并分析了实现复杂度,同时,介绍了基于移位多项式基的伴随式表达式的推导过程,在此基础上提出了一种低复杂度的并行实现结构,并对该结构的复杂度进行了深入分析。最后,采用三种方法设计实现了两种 RS 码型的并行伴随式计算电路,并对实现复杂度进行了分析比较。

2　并行伴随式计算电路

2.1　并行伴随式电路的结构

对于一个 RS(n, k)码(n 为码长,k 为信息位长度),其纠错能力 $t=\lfloor n-k/2 \rfloor$。假设生成多项式 $G(x)$ 如式

$$G(x)=(x-\alpha^0)(x-\alpha^1)\cdots(x-\alpha^{2t-2})(x-\alpha^{2t-1}) \tag{1}$$

所示,式中,α 为本原多项式 $p(x)=x^m+p_{m-1}x^{m-1}+\cdots+p_1x+p_0$ 的本原元;p_m, …, p_0 为本原多项式的系数;m 为有限域的维数。又假设接收到的码字多项式为

$$\begin{aligned} R(x) &= \sum_{i=0}^{n-1} r_i(x)^i \\ &= r_{n-1}x^{n-1}+r_{n-2}x^{n-2}+\cdots r_1x+r_0 \end{aligned} \tag{2}$$

式中,r_j 为有限域 GF(2^m)中的任意元素,可以表示为

$$r_j=b_{m-1}\alpha^{m-1}+b_{m-2}\alpha^{m-2}+\cdots+b_1\alpha-b_0 \tag{3}$$

b_{m-1},…, b_0 为多项式的系数。将式(1)的根代入式(2)就可以得到伴随式 $S_i(1\leqslant i\leqslant 2t)$:

$$S_i=R(\alpha^i)=\sum_{i=0}^{n-1} r_i(\alpha^i)^i \tag{4}$$

与式(4)对应的电路即为串行结构的伴随式计算电路,如图 1 所示。在串行电路中,一个时钟周期只能处理 1 个符号,因此,共需 n 个时钟周期才能完成一个码字的计算。而并行结构则可以有效地减少所需的时钟周期,例如,并行度为 d 时,只需 n/d 个时钟周期就可以完成整个伴随式的计算。文献[4]设计了并行度为 3 的 RS(255,239)并行伴随式计算电路,但给出的电路只能用于码长能整除 3 的码型,不适用于码长不能整除 3 的码型。

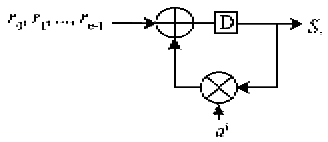

图 1　串行伴随式单元电路

下面将把文献[4]的电路结构从并行度 3 推广到并行度 d。首先分析码长能整除 d 的情况,当码长 n 能整除 d 时,对式(4)进行变换得到式

$$\begin{aligned} S_i &= R(\alpha^i) \\ &= (\cdots(r_{n-1}(\alpha^i)^{d-1}+r_{n-2}(\alpha^i)^{d-2}+\cdots \\ &\quad +r_{n-d+1}\alpha^i+r_{n-d})(\alpha^i)d+\cdots \\ &\quad +r_d)(\alpha^i)^d+r_{d-1}(\alpha^i)^{d-1} \\ &\quad +r_{d-2}(\alpha^i)^{d-2}+\cdots+r_1\alpha^i+r_0 \end{aligned} \tag{5}$$

图 2 所示电路即为与式(5)对应的 d 路并行的伴随式计算单元。在第一个时钟周期,计算 $r_{n-1}(\alpha^i)^{d-1}+r_{n-2}(\alpha^i)^{d-2}+\cdots+r_{n-d+1}\alpha^i+r_{n-d}$,并把计算结果 S_{temp} 送到寄存器中暂存;在第二个时钟周期,计算 S_{temp} 与 $(\alpha^i)^d$ 的积,并在积上加上 $r_{n-d-1}(\alpha^i)^{d-1}+r_{n-p-2}(\alpha^i)^{d-2}+\cdots+r_{n-2d+1}\alpha^i+r_{n-2d}$。接下来的操作与之类似,在最后一个时钟周期,即第 n/d 个时钟周期,把结果存入寄存器,这时寄存器的内容就是伴随式 S_i 的值。

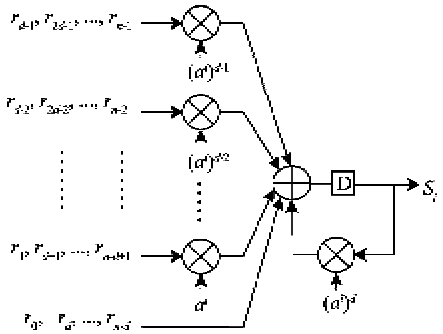

图 2　码长 n 能整除 d 时的并行伴随式单元电路

当码长 n 不能整除并行度 d 时,对式(4)进行变形得到式

$$\begin{aligned} S_i &= R(\alpha^i) \\ &= (\cdots(r_{n-1}(\alpha^i)^{d-1}+r_{n-2}(\alpha^i)^{d-2}+\cdots \\ &\quad +r_{n-d+1}\alpha^i+r_{n-d})(\alpha^i)d+\cdots \\ &\quad +r_{n\bmod d})(\alpha^i)^{n\bmod d}+r_{n\bmod d-1}(\alpha^i)^{n\bmod d-1} \\ &\quad +r_{n\bmod d-2}(\alpha^i)^{n\bmod d-2}+\cdots+r_1\alpha^i+r_0 \end{aligned} \tag{6}$$

其中 $n \bmod d$ 表示 n 关于 d 的余数。

与式(6)对应的电路如图 3 所示。与图 2 相比，该电路增加了一个选择器 MUX，该选择器可以在 $S_{\text{temp}} \times (\alpha^i)^d$ 与 $S_{\text{temp}} \times (\alpha^i)^{n \bmod d}$ 之间选择一路作为加法器的输入，其余部分的操作与图 2 相同。从第一个时钟周期到第 $\lfloor n/d \rfloor$ 个时钟周期，MUX 选择 $S_{\text{temp}} \times (\alpha^i)^d$ 作为加法器的输入。在最后一个时钟周期，MUX 选择 S_{temp} 与 $(\alpha^i)^{n \bmod d}$ 相乘的结果。

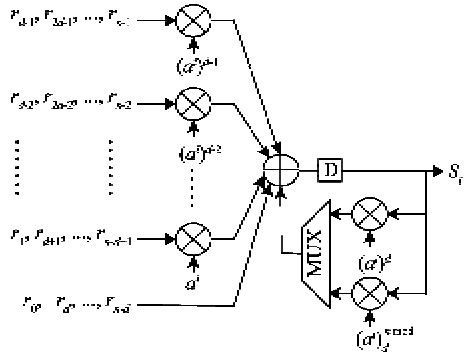

图 3 码长 n 不能整除 d 时的并行伴随式单元电路

可以看出，两种情况下，并行伴随式电路结构基本相同，分别有 $d+1$ 和 $d+2$ 个有限域乘法器，以及一个 $m(d+1)$ 比特输入的有限域加法器，只是后一种情况稍微复杂一点，增加了一个选择器。

2.2 复杂度分析

下面分析并行伴随式计算电路的实现复杂度。从图 2 中可以看出，每次有限域乘法器计算完毕后，有限域加法器将 d 个有限域乘法器的结果及一个输入信号相加就得到了 S_{temp}，可以表示为

$$S_{\text{temp}} = \sum_{j=0}^{d} X_j = X_0 + X_1 + X_2 + \cdots + X_d \quad (7)$$

式中，$X_j = (\alpha^i)^j r_j$。又由有限域的性质有

$$\alpha^m = p_{m-1}\alpha^{m-1} + p_{m-2}\alpha^{m-2} + \cdots + p_1\alpha^1 + p_0 \quad (8)$$

式中，p_m，$\cdots$ p_0 为本原多项式的系数。

先计算 $\alpha \cdot r_j$ 的值，由式(3)有

$$\alpha \cdot r_j = b_{m-1}\alpha^m + b_{m-2}\alpha^{m-1} + \cdots + b_1\alpha^2 + b_0\alpha^1 \quad (9)$$

将式(8)代入式(9)得

$$\begin{aligned}\alpha \cdot r_j &= b_{m-1}(p_{m-1}\alpha^{m-1} + p_{m-2}\alpha^{m-2} + \cdots \\ &\quad + p_1\alpha^1 + p_0) + b_{m-2}\alpha^{m-1} + \cdots \\ &\quad + b_1\alpha^2 + b_0\alpha^1 \\ &= (b_{m-1}p_{m-1} + b_{m-2})\alpha^{m-1} \\ &\quad + (p_{m-2}b_{m-1} + b_{m-3})\alpha^{m-2} + \cdots \\ &\quad + p_0 b_{m-1}\end{aligned} \quad (10)$$

将其表示成矩阵形式：

$$\alpha \cdot r_j = \begin{pmatrix} p_{m-1} & 1 & 0 & \cdots & 0 \\ p_{m-2} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_1 & 0 & 0 & \cdots & 1 \\ p_0 & 0 & 0 & \cdots & 0 \end{pmatrix} \times \begin{pmatrix} b_{m-1} \\ b_{m-2} \\ \vdots \\ b_1 \\ b_0 \end{pmatrix} \quad (11)$$

再将式(11)推广，就可得到 X_j 的表达式如下：

$$\begin{aligned} X_j &= \alpha^{ij} r_j \\ &= \alpha^{ij}(b_{m-1}\alpha^{m-1} + b_{m-2}\alpha^{m-2} + \cdots + b_1\alpha + b_0) \\ &= \begin{pmatrix} p_{m-1} & 1 & 0 & \cdots & 0 \\ p_{m-2} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_1 & 0 & 0 & \cdots & 1 \\ p_0 & 0 & 0 & \cdots & 0 \end{pmatrix}^{ij} \times \begin{pmatrix} b_{m-1} \\ b_{m-2} \\ \vdots \\ b_1 \\ b_0 \end{pmatrix} \end{aligned} \quad (12)$$

式(12)描述的乘法运算就是伴随式计算电路中的核心运算，该乘法器的一个乘数固定，因此被称为有限域常量乘法器，而且乘数为 $(\alpha^i)^j$ 的有限域常量乘法器要比乘数为 α 的乘法器更为复杂。

由于乘法运算和加法运算都可以归结为异或运算，因此，并行伴随式计算电路的复杂度可以用乘法器和加法器中异或门的数量来衡量。对式(12)分析后得知，图 2 中，每个乘法器需要 $m \times m \times \rho$ 个异或门，其中 ρ 为式(12)系数矩阵 α^{ij} 中 1 的密度，即 1 的个数占所有元素的比例。而 $d+1$ 输入的加法器需要 $m \times d$ 个异或门。而纠错能力为 t 的 RS 译码器有 $2t$ 个伴随式计算单元，因此，整个伴随式电路中，乘法器总共有 $2t \times d \times m \times m \times \rho_1$ 个异或门，其中 ρ_1 为所有式(12)所示的系数矩阵中 1 的平均密度；加法器共有 $2t \times m \times d$ 个异或门。可以看到，乘法器和加法器中异或门的数量与并行度 d 成线性关系，且随着 d 的增加而增加。

3 并行伴随式计算电路的结构优化

下面介绍并行伴随式电路的结构优化。通常，减少异或门的数量可以采用文献[6]的迭代匹配算法(IMA)实现，该算法的基本思想是在一个或多个有限域乘法器中寻找能够共享的异或子表达式，以达到减少异或门数量的目的。但是当对多个乘法器采用 IMA 算法进行优化时，过程较为复杂，有时很难找到最优解。而且 IMA 算法只能对乘法器优化，

不能对加法器进行优化,因此,优化效果受到一定限制。我们的优化思路是对式(12)进行适当的变换,使得变换后的结构既适合采用 IMA 算法进行乘法器优化,又能够减少加法器的异或门数。

设

$$\begin{aligned} Y_j &= (\alpha^i)^j r_j \\ &= \alpha^{ij}(b_{m-1}\alpha^{m-1} + b_{m-2}\alpha^{m-2} + \cdots + b_1\alpha + b_0) \end{aligned} \tag{13}$$

整理后得

$$\begin{aligned} Y_j = {} & b_{m-1}\alpha^{m-1+ij} + b_{m-2}\alpha^{m-2+ij} + \cdots \\ & + b_1\alpha^{1+ij} + b_0\alpha^{ij} \end{aligned} \tag{14}$$

若 $\{\alpha^k \mid 0 \leqslant k \leqslant m-1\}$ 被称为 GF(2^m)的多项式基,有序集 $\{\alpha^{k+v} \mid 0 \leqslant k \leqslant m-1\}$ 则被称为相对 $\{\alpha^k \mid 0 \leqslant k \leqslant m-1\}$ 的移位多项式基[8],它是在多项式基上对每一个元素叠加一个移位变量所产生的新基。比较式(12)和式(14)可以看出,Y_j 可以看成以 $\{\alpha^{k+ij} \mid 0 \leqslant k \leqslant m-1\}$ 为基的信号,且其系数与 X_j 在多项式基下的系数相同,都为 $\{b_0, b_1, \cdots, b_{m-1}\}$。

图 4 所示的是与式(14)对应的改进后的伴随式计算单元,它只由一个有限域加法器和一个乘法器组成。由于采用移位多项式,使得图 2 相应位置上的乘法器都省去了,与之相应的是在加法器后面增加了一个乘法器。在第一个时钟周期,经过乘法器后,得到 $r_{n-1}(\alpha^i)^{d-1} + r_{n-2}(\alpha^i)^{d-2} + \cdots + r_{n-d+1}\alpha^i + r_{n-d}$,并把计算结果 S_{temp} 送到寄存器中暂存;在第二个时钟周期,乘法器计算完成 $S_{\text{temp}}(\alpha^i)^d + r_{n-d-1}(\alpha^i)^{d-1} + r_{n-\rho-2}(\alpha^i)^{d-2} + \cdots + r_{n-2d+1}\alpha^i + r_{n-2d}$。接下来的操作与之类似,在最后一个时钟周期,即第 n/d 个时钟周期,把结果存入寄存器,这时寄存器的内容就是伴随式 S_i 的值。

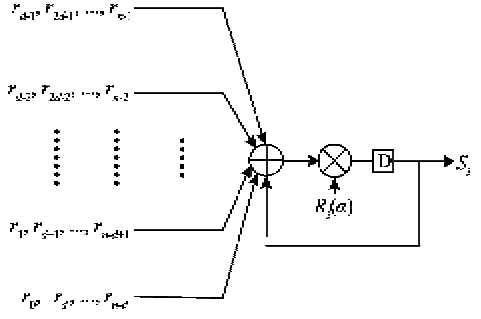

图 4　改进的并行伴随式单元结构

下面分析图 4 电路的复杂度。对于加法器而言,由于计算 S_{temp} 需要把用不同基表示的数加在一起,因此,要做适当的变化。如图 5 所示,由于加法运算实际上是异或运算,只需把重叠部分基的系数相加即可。当 $1 \leqslant i \leqslant m$ 时,重叠部分有 $m + di$ 比特,得到 $m + di$ 比特的和;当 $m < i$ 时,任意两个基之间没有重叠部分,不需要任何运算,可以直接输出。这样,对于所有 $2t$ 个伴随式单元,加法器总共需要 $(m-1) \times m/2 \times d$ 个异或门。很明显,图 4 的加法器的复杂度比图 2 大为降低。

与加法器相接的乘法器完成的是从移位多项式基到多项式基的转换,即将 $m + di$ 比特或 $dm + m$ 比特的和映射到 m 比特。由于不同的伴随式计算单元使用不同的移位基,使得该乘法器的系数矩阵也互不相同,具体的系数矩阵可以参考公式(15)。对于所有 $2t$ 个伴随式单元,乘法器总共需要 $[-dm^2 + (4td + 4t + d)m - 4t] \times m/2 \times \rho_2$ 个异或门,其中 ρ_2 为式

$$Co = \begin{cases} (\alpha^0 \ \ \alpha^1 \ \cdots \ \alpha^{di+m-1}), & 1 \leqslant i \leqslant m \\ (\alpha^0 \ \cdots \ \alpha^{m-1} \mid \alpha^i \ \cdots \mid \alpha^{di} \ \cdots \ \alpha^{di+m-1}), & i > m \end{cases} \tag{15}$$

系数矩阵中 1 的平均密度。

4　实验结果

本文以 RS(2040, 2024)和 RS(255, 239)的并行伴随式计算电路为例,对直接实现法、IMA 法和本文方法三种实现方法进行了研究。直接法采用图 2 结构,没有对电路进行任何优化,IMA 法则对图 2 中的有限域乘法器用IMA算法进行优化,本文方法则采

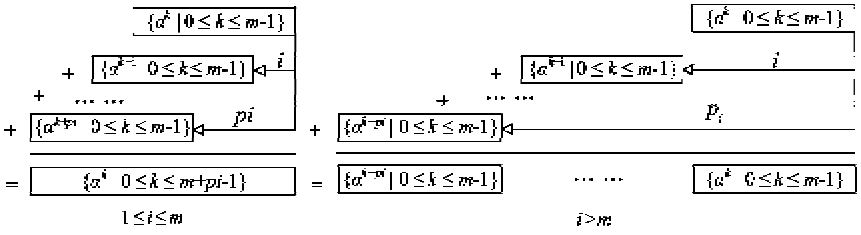

图 5　改进结构中的加法器

用图 4 的改进结构,同时对电路中的乘法器也用 IMA 算法进行优化。实现复杂度用异或门门数来度量,而异或门门数受到综合工具和约束条件的影响,所以直接用上文中复杂度的计算公式来计算异或门门数。

表 1 和表 2 分别为并行度为 8 和 64 时 RS(2040,2024)并行伴随式计算电路复杂度的比较结果。从表 1 可以看出,当 $d = 8$ 时,IMA 法中乘法器的复杂度比直接实现节省了 26%,但由于 IMA 法不能对加法器进行优化,所以总的异或门数只减少了 18%。本文方法中,乘法器的复杂度与 IMA 方法相当,但是加法器却减少了 69%,因此,总的异或门数比前两种方法分别减少了 29%和 41%。表 2 中,当 $d = 64$ 时,IMA 法使得乘法器的异或门比直接实现节省了 38%,但加法器没有优化,总的异或门减少了 31%。本文方法中的乘法器比 IMA 法减少了 40%,同时加法器也减少了 69%,因此,总的异或门数比 IMA 法减少了 64%。从上面的分析可以看出,随着并行度的增加,本文方法和 IMA 法的优化效果也更加明显,但本文方法优于 IMA 法。

表 1 RS(2040,2024)并行伴随式计算电路资源比较($d = 8$)

实现方法		异或门数量	与[A]比较	与[B]比较
直接实现法[A]	乘法器	3110	/	/
	加法器	1408	/	/
	总计	4518	/	/
IMA 法[B]	乘法器	2291	26%	/
	加法器	1408	0%	/
	总计	3699	18%	/
本文方法	乘法器	2204	29%	5%
	加法器	440	69%	69%
	总计	2644	41%	29%

表 2 RS(2040,2024)并行伴随式计算电路资源比较($d = 64$)

实现方法		异或门数量	与[A]比较	与[B]比较
直接实现法[A]	乘法器	45796	/	/
	加法器	11264	/	/
	总计	57060	/	/
IMA 实现[B]	乘法器	28186	38%	/
	加法器	11264	0%	/
	总计	39450	31%	/
本文方法	乘法器	16872	63%	40%
	加法器	3520	69%	69%
	总计	20392	64%	48%

表 3 和表 4 分别为并行度为 8 和 64 时 RS(255,239)并行伴随式计算电路复杂度的比较结果。从表 3可以看出,当 $d = 8$ 时,本文方法中的乘法器的复杂度与 IMA 方法相当,但是加法器却减少了 78%,因此,总的异或门数比前两种方法分别减少了 47%和 30%。表 4 中,当 $d = 64$ 时,本文方法中的乘法器比 IMA 方法减少了 35%,同时加法器也减少了 78%,总的异或门数比 IMA 法减少了 50%。比较 RS(2040,2024)和 RS(255,239)的实验结果可以看出,对于不同的 RS 码型,本文方法都能取得较好的优化效果。

表 3 RS(255,239)并行伴随式计算电路资源比较($d = 8$)

实现方法		异或门门数	与[A]比较	与[B]比较
直接实现[A]	乘法器	2744	/	/
	加法器	1024	/	/
	总计	3768	/	/
IMA 实现[B]	乘法器	1817	34%	/
	加法器	1024	0%	/
	总计	2841	25%	/
本文方法	乘法器	1754	36%	3%
	加法器	224	78%	78%
	总计	1978	47%	30%

表 4 RS(255,239)并行伴随式计算电路资源比较($d = 64$)

实现方法		异或门门数	与[A]比较	与[B]比较
直接实现法[A]	乘法器	24270	/	/
	加法器	8192	/	/
	总计	32462	/	/
IMA 法[B]	乘法器	15097	38%	/
	加法器	8192	0%	/
	总计	23289	28%	/
本文方法	乘法器	9836	59%	35%
	加法器	1792	78%	78%
	总计	11628	64%	50%

图 6 和图 7 分别给出了 RS(2040, 2024)伴随式计算电路中加法器和乘法器的实现复杂度与并行度 d 的关系。可以看出,加法器和乘法器的复杂度均随着并行度增加而线性增加,但是本文方法的增加速度明显低于 IMA 法,这说明随着并行度的增大,本文结构可以获得更多的优化。

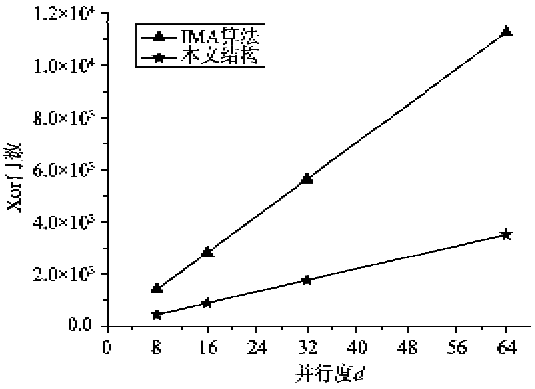

图 6　不同并行度 d 的加法器复杂度比较

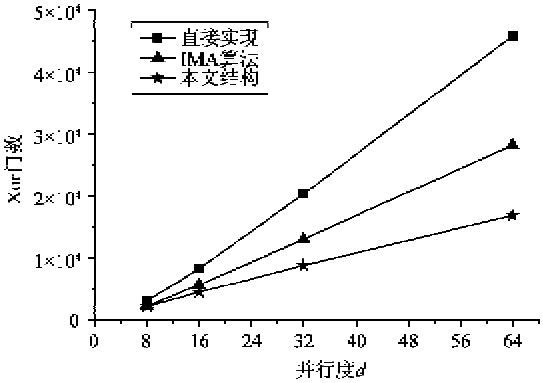

图 7　不同并行度 d 的乘法器复杂度比较

图 8 和图 9 分别描述了并行度为 16 的 RS(2040, k, t)伴随式电路中加法器和乘法器的复杂度随纠错能力 t 的变化情况。从图 8 可以看出,采用 IMA 法,加法器的复杂度随着 t 线性增加,而本文方法中,加法器复杂度与 t 无关,保持在一个较低的数量不变。这时由于图 2 结构中,加法器的异或门数量与 t 成线性关系,而 IMA 法只能优化乘法器,不能优化加法器。而改进结构中,参与加法运算的操作数的基没有重叠部分,所以不需要额外的逻辑资源。图 9 则清楚表明,当纠错能力变化时,采用本文方法和 IMA 法,乘法器的复杂度均明显低于直接实现法,且本文方法优于 IMA 法。

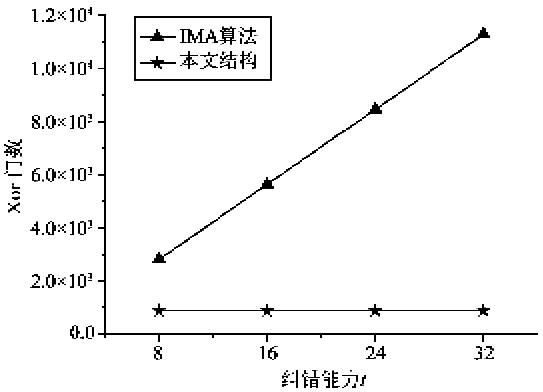

图 8　不同纠错能力 t 的加法器复杂度比较

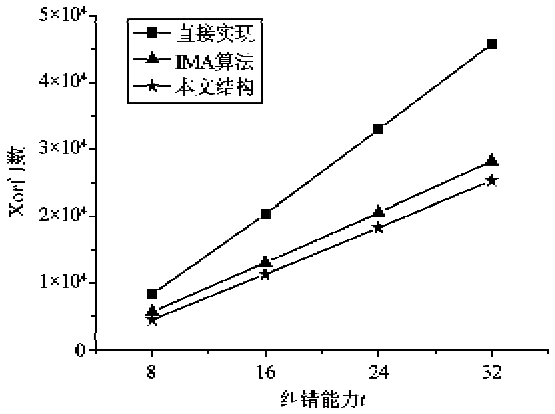

图 9　不同纠错能力 t 的乘法器复杂度比较

通过以上比较,可以看出改进结构可以大大节省逻辑资源,但是由于改进结构的乘法器采用 IMA 方法来优化,而 IMA 的优化程效果依赖于乘法矩阵的大小,当码长变小时,乘法器的优化性能就会变差。由于加法器采用图 5 的结构,与码长无关,其优化效果不受码长的影响。

5　结　论

本文研究了并行伴随式计算电路的通用设计方法和优化结构。针对码长与并行度的关系,给出了两种电路结构来实现并行伴随式计算。为了降低并行伴随式计算电路的复杂度,提出了一种改进结构。在改进结构中,对乘法器进行改进,省去了加法器输入端的多个乘法器,从而降低了乘法的复杂度;改进加法器的结构,只对有重叠部分的基相加,使得加法大大简化。实验结果表明对于 RS(2040,2024)该结构比直接实现能节约 64%的逻辑资源,而且在并行度和纠错能力变化时,仍然可以获得较好的优化结果。

参考文献

[1] Baek J H, Sunwoo M H. New degree computationless modified Euclid algorithm and architecture for Reed-Solomon decoder. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2006, 14: 915-920

[2] Seungbeom L, Hanho L, Jongyeon S, et al. A high-speed pipelined degree-computationless modified Euclidean algorithm architecture for Reed-Solomon decoders. In: Proceedings of

the 2007 IEEE International Symposium on Circuits and Systems (ISCAS 2007), New Orleans, USA, 2007. 901-904

[3] Junho C, Wonyong S. Strength-Reduced parallel Chien search architecture for strong BCH codes. *IEEE Transactions on Circuits and Systems* part Ⅱ: Express Briefs, 2008, 55(5): 427-431

[4] Song L, Yu M L, Shaffer M S. 10 and 40-Gb/s forward error correction devices for optical communications, *IEEE Journal of Solid-State Circuit*, 2002, 37(11): 1565-1573

[5] Lee S, Choi C, Lee H. Two-parallel Reed-Solomon based FEC architecture for optical communications. *IEICE Electronics Express*, 2008, 5(10): 374-380

[6] Chen Y, Parhi K K. Small area parallel chien search architectures for long BCH codes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2004, 12: 545-549

[7] Hu Q, Wang Z, Zhang J, et al. Area optimization of parallel Chien search architecture for Reed-Solomon (255, 239) decoder. *Journal of Southeast University (English Edition)*, 2005, 22(1): 5-10

[8] Fan H, Dai Y. Fast bit-parallel GF(2n) multiplier for all trinomials. *IEEE Transactions on Computers*, 2005, 54: 485-490

Optimal design of Reed-Solomon parallel syndrome computation with shifted polynomial basis

Zhang Liang, Wang Zhigong, Hu Qingsheng

(Institute of RF- & OE-ICs, Southeast University, Nanjing 210096)

Abstract

The research on optimization of Reed-Solomon (RS) parallel syndrome computation was conducted. The expressions for parallel syndrome computation when the codeword length is and is not exactly divided by the parallel factor were derived, and their corresponding circuits were designed. In consideration of the circuit complexity increment caused by the parallel design, an improved parallel syndrome computation architecture with the reduced complexity was proposed by transforming the syndrome expression properly and representing the production of the multiplier, in the shifted polynomial basis. In this architecture, the complexity of the multiplications is reduced by removing the multipliers in the inputs of the adders, and the complexity of the additions is diminished by decreasing the overlapped basis. The experimental results show that the hardware complexity can be reduced by 40% in the design of the RS(2040,2024) and RS(255,239) codes with the parallel factor of 8, and the hardware complexity can be reduced by 64% in the design of the RS(2040, 2024) codes with the parallel factor of 64.

Key words: Reed-Solomon (RS) decoder, parallel syndrome computation, shifted polynomial basis, low-cost architecture