并行 BCH 伴随式计算电路的优化

张 亮 王志功 胡庆生
(射频与光电集成电路研究所, 东南大学 南京 210096)

摘 要: 随着通信系统的速率越来越高, 对 BCH 译码器吞吐量的要求也不断提高。由于 BCH 码是串行的处理数据, 在吞吐量大的应用时一般需要并行处理, 但这会导致电路的复杂度显著增加。本文主要研究并行伴随式计算电路的优化。通过合并输入端的常量乘法器, 得到改进的并行伴随式结构。该结构克服了传统方法只能对局部的乘法器进行优化的缺点, 可以对全部乘法器进行优化, 从而有效的减少逻辑资源。实验结果表明, 对于并行度为 64 的 BCH(2040, 1952)译码器, 本文的优化结构可以节省 67% 的逻辑资源, 而且在并行度、纠错能力和码长变化时, 仍然可以获得较好的优化结果。

关键词: 伴随式计算电路; 并行处理; BCH 码
中图分类号: TN492 **文献标识码**: A **文章编号**: 1003 - 0530(2010)03 - 0458 - 04

Optimization of parallel syndrome computation for BCH codes

ZHANG Liang WANG Zhi-gong HU Qing-sheng
(Institute of RF-& OE-IC, Southeast University, Nanjing, China)

Abstract: Due to the increasing demand for high capacity of communications, Bose-Chaudhuri-Hochquenghem decoders with high throughput are desirable to meet higher data rate. Since the BCH codes conduct the bit-by-bit error correction, they often need a parallel implementation for high throughput application. In this paper, we propose the optimization for the parallel syndrome computation architecture. The improved architecture is proposed by simplifying the multipliers of the input. In the improved architecture, all the multipliers can be optimized, while only part of the multiplier can be optimized by the traditional methods. The experimental results show that the hardware complexity can be reduced by 67% in the design of the BCH(2040,1952) codes with parallel factor 64. In different cases of parallel factors, error correcting capability and codeword length, the improved architecture also have good performance compared with the traditional methods.

Key words: Bose-Chaudhuri-Hochquenghem (BCH) codes; parallel processing; syndrome computation

1 引言

BCH(Bose-Chandhari-Hocquenghem)码具有很好的纠错能力,广泛应用于各种系统中,如磁盘记录系统、固态存储系统、无线通信系统以及光通信系统。由于 BCH 译码器对数据进行串行的处理,吞吐量比较小。对于高吞吐量的应用场合,如光通信系统和存储系统,一般需要并行的 BCH 译码器来满足吞吐量的要求。

基于有限域 GF(2^m)的 BCH(n, k, t),码长 $n = 2^m - 1$,信息位长 $k = n - mt$,纠错能力为 t。BCH 译码器包括以下三个单元:伴随式计算、解关键方程和钱氏搜索。伴随式计算单元完成从接受的码字中计算出伴随式;解关键方程利用伴随式来获得错误位置多项式;钱氏搜索单元采用依次把所有错误位置带入错误位置多项式的方式来获得错误位置。在这三个单元中,低复杂度的解关键方程单元和并行钱氏搜索研究的比较多[1-4],而且取得了较好的性能。而很少有文献涉及到并行伴随式计算电路优化。

由于并行结构的复杂度大大增加,所以降低实现复杂度是并行设计中需要考虑的重要问题。在伴随式计算中主要的运算是有限域上的乘法运算,有很多文献涉及有限域运算的优化。文献[5]提出了迭代匹配算法(Iterative Matching Algorithm, IMA)和组匹配算法(Group Matching, Algorithm, GMA)来优化乘法运算,这两种算法分别针对单个系数矩阵和多个系数矩阵采用贪婪算法来寻找可以合并的子式从而降低乘法的复杂度。文献[6]提出了全局优化算法,用来寻找全局最优的重复子式来减少乘法的复杂度。但这些算法只适用于输入为多比特的乘法器的优化,不适合输入为单比特的乘法器的优化,而并行伴随式计算电路中的

收稿日期: 2009 年 6 月 15 日; 修回日期: 无

乘法器大部分为单比特乘法器,所以直接应用这些算法并不能取得很好的优化结果。

本文主要研究并行伴随式计算电路的优化设计。第 2 部分研究了伴随式计算电路的并行实现方法。第 3 部分介绍低复杂度伴随式计算电路。第 4 部分给出了实验结果,并与现有的算法作出比较。在第 5 部分对全文进行总结。

2　并行伴随式电路的直接实现

通过把生成多项式 $G(x)$的根代入码字多项式 $R(x)$中,就得到了所有的伴随式 $S_i(1\le i\le 2t)$,见式(3)。可以根据式(3)可以直接得到串行伴随式计算电路,如图 1 所示。由于串行结构一个时钟周期只能处理 1 位数据,共需 n 个时钟周期才能完成所有数据的计算。

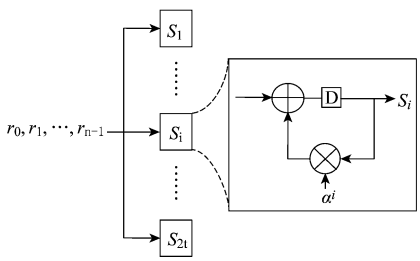

图 1　串行伴随式计算电路

$$R(x)=\sum_{i=0}^{n-1} r_i(x)^i = r_{n-1}x^{n-1}+r_{n-2}x^{n-2}+\cdots r_1x+r_0 \quad (1)$$

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

$$S_i=R(\alpha^i)=\sum_{i=0}^{n-1} r_i(\alpha^j)^i \quad (3)$$

文献[7]设计了并行度为 3 的并行伴随式计算电路,文献[8]给出了并行度为 2 的并行伴随式计算电路,但没有文献给出并行伴随式计算电路的一般形式,下面我们把文献[7]的电路从并行度 3 推广到并行度 p。对式(3)进行变形得到式(4),与其对应的并行伴随式单元结构如图 2 所示。由于每个时钟周期可以处理 p 比特数据,则只需 n/p 个时钟周期就可完成计算。在第一个时钟周期,计算 $r_{n-1}(\alpha^i)^{p-1}+r_{n-2}(\alpha^i)^{p-2}+\cdots+r_{n-p+1}\alpha^i+r_{n-p}$,并把计算结果 S_{temp}送到寄存器暂存。在第二个时钟周期,计算 S_{temp}与$(\alpha^i)^p$ 的积,并在积上加上 $r_{n-p-1}(\alpha^i)^{p-1}+r_{n-p-2}(\alpha^i)^{p-2}+\cdots+r_{n-2p+1}\alpha^i+r_{n-2p}$,类似地,在第 n/p 个时钟周期,计算 $r_{p-1}(\alpha^i)^{p-1}+r_{p-2}(\alpha^i)^{p-2}+\cdots+r_1\alpha^i+r_0$,并把新的结果存入寄存器,这时,寄存器的内容就是伴随式 S_i。

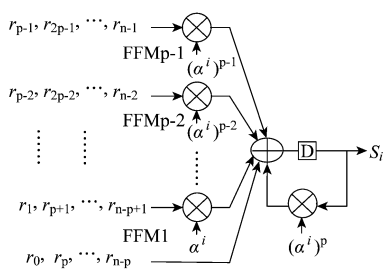

图 2　并行伴随式的单元电路

$$S_i=R(\alpha^i)=(\cdots(r_{n-1}(\alpha^i)^{p-1}+r_{n-2}(\alpha^i)^{p-2}+\cdots+r_{n-p+1}\alpha^i+r_{n-p})(\alpha^i)^p+\cdots+r_p)(\alpha^i)^p+r_{p-1}(\alpha^i)^{p-1}+r_{p-2}(\alpha^i)^{p-2}+\cdots+r_1\alpha^i+r_0 \quad (4)$$

IMA 算法[5]可以用来优化有限域乘法器,该算法利用乘法系数矩阵各列之间的相关位,来寻找可以合并的子式,从而降低乘法的复杂度。所以使用 IMA 算法时,乘法器系数矩阵的列数必须大于 1,即输入必须为多比特。在图 2 的结构中,乘法器 FFM1, FFM2…FFMp-1 的输入为单比特数据,此时乘法的系数矩阵的维数为 $m\times 1$,只有 1 列,不能用 IMA 算法来优化。乘法器 FFMp 的输入为 m 比特,乘法的系数矩阵为 $m\times m$,所以 IMA 算法只能用来优化乘法器 FFMp。下面给出了一种改进的并行伴随式电路结构,可以对所有的乘法器进行优化。

3　并行伴随式计算的改进结构

对于基于 GF(2^m)的 BCH(n,k,t)的有限域常量乘法器,每个乘法器位宽为 m 比特,则 m 比特的变量 B 和 m 比特的常数 α^j相乘的结果可以表示为:

$$\begin{aligned} X_j &= \alpha^j B \\ &= \alpha^j(b_0+b_1\alpha+b_2\alpha^2+\cdots+b_{m-1}\alpha^{m-1}) \\ &= \begin{pmatrix} a_0^j & a_0^{j+1} & \cdots & a_0^{j+m-1} \\ a_1^j & a_1^{j+1} & \cdots & a_1^{j+m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1}^j & a_{m-1}^{j+1} & \cdots & a_{m-1}^{j+m-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{pmatrix} \\ &= X_{j,0}+X_{j,1}\alpha+X_{j,2}\alpha^2+\cdots+X_{j,m-1}\alpha^{m-1} \end{aligned} \quad (5)$$

对于输入为单比特的有限域乘法器,由于输入为单比特变量 b,乘法器的系数矩阵为 $m\times 1$,乘法过程可以表示为:

$$X_j=\alpha^j b = \begin{pmatrix} a_0^j \\ a_1^j \\ \vdots \\ a_{m-1}^j \end{pmatrix} \cdot b \quad (6)$$

假设在并行伴随式的单元电路中输入的 p 比特信号为$\{b_0,b_1,b_2,\cdots,b_{p-1}\}$,可以把乘法器 FFM1, FFM2, …,FFMp-1 输出之和表示为下面的形式:

$$\begin{aligned} Y_j &= b_0+b_1\alpha^i+\cdots+b_{p-2}(\alpha^i)^{p-2}+b_{p-1}(\alpha^i)^{p-1} \\ &= \begin{pmatrix} a_0^0 & a_0^i & \cdots & a_0^{i(p-1)} \\ a_1^0 & a_1^i & \cdots & a_1^{i(p-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1}^0 & a_{m-1}^i & \cdots & a_{m-1}^{i(p-1)} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{pmatrix} \end{aligned}$$

$$= Y_{j,0} + Y_{j,1}\alpha + Y_{j,2}\alpha^2 + \cdots + Y_{j,m-1}\alpha^{m-1} \quad (7)$$

根据式(7)和式(5),可以把图 2 的并行伴随式计算单元输入端的乘法器 FFM1, FFM2,…, FFMp－1 简化为一个乘法器,从而得到改进的并行伴随式计算单元电路,如图 3 所示。p 比特的输入码字与乘法器 FFMA 相乘后,输出位宽为 m 比特的积,同时乘法器 FFMB 也输出位宽为 m 比特的积,这两个积在有限域加法器 FFA 相加,得到的和送入寄存器暂存。

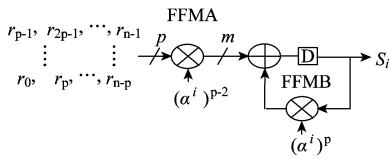

图 3 改进的并行伴随式计算单元电路

图 3 描述的是并行伴随式计算单元电路,对于纠错能力为 t 的 BCH 译码器需要 $2t$ 个单元电路。每个单元中乘法器 FFMA 的输入都相同,可以考虑把 $2t$ 个单元中所有的 FFMA 乘法器进行合并,合并后的结果可以表示为:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_{2t} \end{pmatrix} = \begin{pmatrix} a^0 & a^1 & \cdots & a^{p-1} \\ a^0 & a^2 & \cdots & a^{2(p-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a^0 & a^{2t} & \cdots & a^{2t(p-1)} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{p-1} \end{pmatrix} \quad (8)$$

根据式(8),对图 3 的单元结构进行合并,得到新的结构,如图 4 所示。用乘法器 FFMA 来代替传统结构中的 $2t \times (\rho - 1)$个乘法器,且乘法器 FFMA 的系数矩阵如式(8)所示。p 比特的输入码字经过乘法器 FFMA 后输出位宽为 $2tm$ 比特的积,按 m 比特为一组对积进行分组,依次把各组送到 $2t$ 个加法器,每个加法器完成对相应伴随式的累加。在图 4 中,乘法器 FFMA 的系数矩阵维数为 $p \times 2tm$,而其他的乘法器维数为 $m \times m$,所有系数矩阵的列数都大于 1,这样就可以对所有的乘法器进行优化。

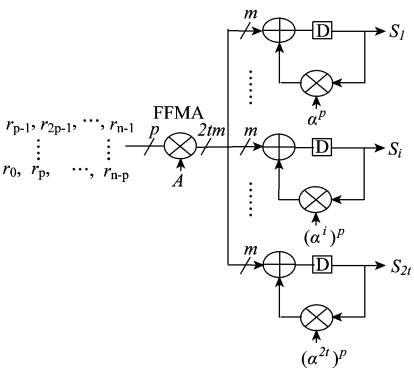

图 4 改进的并行伴随式计算电路

4 实验结果

我们分别比较了三种算法的复杂度:直接实现、IMA 实现和本文的结构。直接实现是图 2 的直接实现,没有对图中的乘法器进行优化;IMA 实现是对图 2 中的有限域乘法器用 IMA 算法[5]进行优化,IMA 对每个乘法器的系数矩阵进行优化,找出其中能合并的子项来降低实现复杂度。本文的结构中乘法器的系数矩阵的规模分别为:1) $p \times 2tm$; 2) $m \times m$,可以对所有乘法器进行 IMA 优化。我们以 BCH(2040, 1952)并行伴随式计算为例,来比较不同方法的复杂度差异,表 1 和表 2 比较了并行度分别为 8 和 64 时的复杂度。

对于并行度为 8 的并行伴随式计算单元,IMA 实现的复杂度比直接实现节省了 19%,而本文结构比 IMA 实现和直接实现分别节省了 39% 和 51%。对于并行度为 64 的并行伴随式计算单元,IMA 实现所用的异或门数比直接实现节省了 5%,本文结构比 IMA 实现和直接实现分别节省了 65% 和 67% 的逻辑资源。如果比较归一化面积,本文结构用串行结构 14 倍的复杂度就可以实现 64 倍的吞吐量。

表 1 并行度为 8 的 BCH(2040,1952)并行伴随式计算

实现方法		异或门门数	归一化面积	与[A]比较	与[B]比较
串行结构		134	1	/	/
并行	直接实现[A]	855	6	/	/
	IMA 实现[B]	690	5	19%	/
	本文结构	422	3	51%	39%

表 2 并行度为 64 的 BCH(2040,1952)并行伴随式计算

实现方法		异或门门数	归一化面积	与[A]比较	与[B]比较
串行结构		134	1	/	/
并行	直接实现[A]	5828	43	/	/
	IMA 实现[B]	5512	41	5%	/
	本文结构	1934	14	67%	65%

图 5 描述了 BCH(2040, 1952)伴随式计算的实现复杂度与并行度 p 的关系。直接实现和 IMA 算法中的复杂度随着并行度的增加而线性增长,而改进结构复杂度的斜率随着并行度的增加而减少。这说明随着并行度的增大,本文结构可以获得更多的优化。由于本设计中乘法器的规模与并行度有关,当并行度增大时,乘法器系数矩阵的列数增加,这时候可以找到更多的可以共享的子项,所以能获得更大的优化。

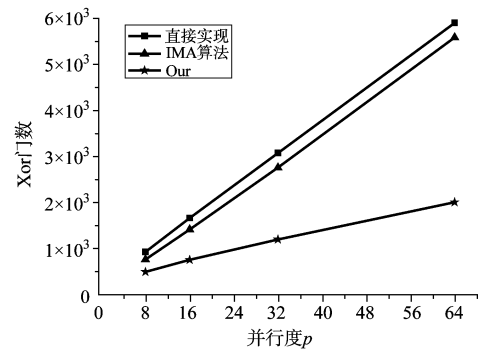

图 5 具有不同并行度 p 时复杂度的比较

图 6 描述了伴随式计算电路的复杂度随纠错能力的变化关系。对于并行度为 16 的 BCH(2040,k,t)译码器,纠错能力 t 分别取 8、16、24 和 32,计算不同纠错能力下的复杂度。从图 6 可以看出,直接实现和 IMA 实现中的复杂度随着 t 线性增加,而改进结构的乘法复杂度曲线一直在 IMA 算法和直接实现的下方,说明在纠错能力变化时,本文的结构仍能取得较好的性能。

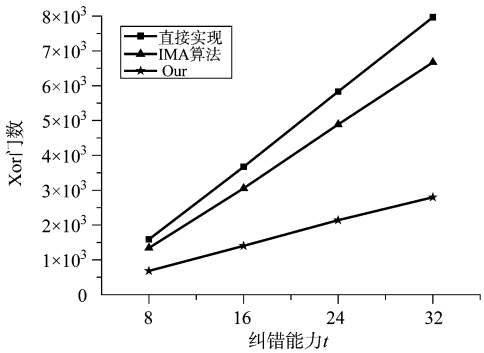

图 6　不同纠错能力 t 时复杂度的比较

表 3 描述了不同码长条件下伴随式计算电路复杂度的变化情况。对于并行度为 16 的 BCH(n,k,8)译码器,码长 n 分别取 1023、2047、4095 和 8191,即有限域为 GF(2^{10})、GF(2^{11})、GF(2^{12})和 GF(2^{13}),计算不同码长下复杂度的变化情况。从表 3 可以看出,本文的结构至少可以比直接实现和 IMA 实现节约 57% 和 46% 的复杂度,这说明本文的结构在码长变化时仍能取得较好的性能。

表 3　不同码长条件下各种算法的比较

码长	直接实现[A]	IMA[B]		改进结构		
		Xor 门数	与[A]比较	Xor 门数	与[A]比较	与[B]比较
1023	1486	1280	14%	608	59%	53%
2047	1589	1341	17%	684	57%	49%
4095	2087	1714	18%	866	59%	49%
8191	2424	1930	20%	1041	57%	46%

5　结论

本文研究了并行伴随式计算电路设计的优化。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,加法器输入端的多个乘法器进行合并,从而降低了实现复杂度。对于并行度为 64 的并行 BCH(2040, 1952),实验结果表明,本文结构所需的逻辑资源比 IMA 实现和直接实现分别节省了 65% 和 67%,而且在并行度、纠错能力和码长变化时,仍然可以获得较好的优化结果。

参考文献

[1] Baek J. H., and M. H. Sunwoo, "New Degree Computationless Modified Euclid Algorithm and Architecture for Reed-Solomon Decoder," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 14, pp. 915-920,2006.

[2] L. Seungbeom, L. Hanho, S. Jongyoon, and K. Je-Soo, "A high-speed pipelined degree-computationless modified euclidean Algorithm Architecture for Reed-Solomon Decoders," 2007 IEEE International Symposium on Circuits and Systems (ISCAS 2007),pp. 901-904,2007.

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

[4] L. Hanho, "A high-speed low-complexity Reed-Solomon decoder for optical communications," Circuits and Systems II: Express Briefs, IEEE Transactions on, vol. 52, pp. 461-465,2005.

[5] Chen Y., and K. K. Parhi, "Small Area Parallel Chien Search Architectures for Long BCH Codes," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 12,pp. 545-549,2004.

[6] Hu Q., Wang Z., and et al., "Area Optimization of Parallel Chien Search Architecture for Reed-Solomon (255, 239) Decoder," Journal of Southeast University (English Edition), vol. 22,no. 1,pp. 5-10,2005.

[7] L. Song, M. L. Yu, and M. S. Shaffer, "10 and 40-Gb/s forward error correction devices for optical communications," IEEE J. Solid-State circuit, vol. 37, no. 11, pp. 1565-1573, Nov,2002.

[8] Seungbeom Lee, Chang-SeoK Choi, and Hanho Lee, "Two-parallel Reed-Solomon based FEC architecture for optical communications," IEICE Electronics Express, vol. 5, no. 10, pp. 374-380,2008.

作者简介

张　亮,男,东南大学博士生,主要从事大规模集成电路设计的研究。

王志功,男,教授,博士生导师,教育部长江学者特聘教授,主要从事超高速、微波、射频、光电和大规模集成电路的研究。

胡庆生,女,教授,博士生导师,主要从事大规模集成电路设计的研究。