BCH 编译码器在 NAND Flash 控制器中的应用研究

郭 鹏[1,2,3],房 亮[3],于沛玲[3]
(1. 中国科学院 光电研究院,北京 100094;
2. 中国科学院大学,北京 100190;
3. 中国科学院 空间应用工程与技术中心,北京 100094)

摘 要:针对空间应用对固态存储器中 ECC 校验在计算速度和纠错能力上的要求,提出了一种应用在 NAND Flash 控制器中的高速并行 BCH 编译码器。文中采用了一种独特的译码器架构,并改进了计算伴随式的算法,先利用编码电路计算出伴随多项式,再利用译码电路计算出伴随式。与直接计算出伴随式的译码器相比,虽然译码时间略有增加,但却能明显减少资源的占用量。结合采用其他一些节省资源和提高运行速度的措施,使该译码器的设计更适应空间应用的需要。

关键词:BCH 码;并行;FFM;伴随多项式;伴随式;NAND Flash 控制器
中图分类号:TP391 **文献标识码**:A **文章编号**:1673-629X(2014)01-0179-05
doi:10.3969/j.issn.1673-629X.2014.01.046

Application and Research of BCH Encoder/Decoder in NAND Flash Controller

GUO Peng[1,2,3], FANG Liang[3], YU Pei-ling[3]
(1. Institute of Opto-electronics, Chinese Academy of Sciences, Beijing 100094, China;
2. University of Chinese Academy of Sciences, Beijing 100094, China;
3. Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China)

Abstract: Based on the requirements of operating rate and correction capability for ECC of solid state memory for space application, a new architecture of parallel BCH encoder and decoder applied in NAND Flash Controller is presented. Design a new architecture of BCH decoder, and improve syndrome calculation algorithm. It utilizes encoder to calculate out syndrome polynomial first, and then uses decoder to calculate out the syndrome. Compared with other decoder which directly calculates out the syndrome, although it has a little decoding time increasing, could significantly reduce the usage of resources. This design combined with some other measures to save resources and improve speed, could better meet the demands of space application.

Key words: BCH code; parallel; Finite Field Multiplier; syndrome polynomial; syndrome; NAND Flash Controller

1 概 述

NAND Flash 是目前应用最广泛的一种非易失性存储介质,具有容量密度大、体积小、功耗低、成本低、抗震动、温湿适应范围宽等特点。它广泛应用于数码相机、固态硬盘等电子产品上,并且以 NAND Flash 为存储介质的固态硬盘也已成为航天、航空及军工领域存储设备的主流设计。这些应用环境普遍较为恶劣,如在航天应用中,芯片易受到宇宙中高能带电粒子的影响,产生单粒子翻转,使存储数据出错。因此,设计面向高可靠应用的固态存储器时,需要使用功能强大的错误校验机制。基于 Flash 的存储器在航天领域的应用还有一些独特之处,比如较大的连续数据吞吐量,更小的资源占用量和更低的功耗要求,因此对相应的错误校验机制也提出了一些独特的要求。

收稿日期:2013-03-14 **修回日期**:2013-06-22 **网络出版时间**:2013-11-12
基金项目:总装备部预研计划项目(xxxx04013205)
作者简介:郭 鹏(1982-),男,河北邢台人,硕士,研究方向为空间电子技术;房 亮,高级工程师,研究方向为空间科学与应用;于沛玲,研究员,研究方向为空间科学与应用。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20131112.1653.058.html

现在常用的对数据校验的机制有奇偶校验、CRC 校验、汉明码、BCH 码校验等。其中,汉明码常用于对 SLC NAND Flash 的数据校验,但其只能实现单比特的纠错。在具体的空间应用中,由于受空间环境的影响,Flash 芯片可能在同一页上会产生多个比特的错误,只能实现单比特纠错的汉明码无法满足应用需求,在一些设计中通过采用 BCH 编译码技术,可以实现多比特位纠错,最大程度上降低误码率。

文献[1-2]中提出了两种类似的 BCH 编译码器设计,均采用了并行编码和简化的无求逆 BM 译码算法。文献[3]中提出了多种并行 Chien 搜索算法架构,提高了译码速度。文献[4]中通过采用分组预取译码、引入流水线操作来提升 BCH 码译码速度。文中借鉴了上述文献中的设计优点,采用并行编码、简化的无求逆 BM 译码算法、并行的 Chien 搜索算法和流水线操作,并提出了在译码电路中复用编码电路的独特架构和相应的伴随式计算算法,进一步减少了编译码器的资源占用量,以适应航天领域应用要求。

2 BCH 编译码算法原理简述

BCH 码属于线性循环码的一种,可以纠检错多个随机错误,码长可变。BCH 码构建在有限域 GF(2^{13})上,码长为 $n = 2^m - 1$ 的 BCH 码被称为本原 BCH 码,在实际编码时通常使用本原 BCH 码的缩码。同其他循环码编码方式相同,BCH 编码的数学计算公式如式(1)和式(2)[5]:

$$r(x) = x^{n-k}m(x) \bmod g(x) \tag{1}$$

$$c(x) = r(x) + x^{n-k}m(x) \tag{2}$$

其中,n 为码长;k 为信息位长度;$c(x)$为码字多项式;$m(x)$为信息位多项式;$g(x)$为生成多项式。对于可纠错数为 t 的 BCH 码,$g(x)$的最高次幂为 $n-k=mt$,所以码长不同,纠错数不同,所对应的生成多项式也就不同。根据文献[6-8]中对分圆陪集和 BCH 码的关系的描述,得到计算生成多项式的公式(3)(a_r 为分圆陪集所对应的有限域元素):

$$g(x) = (x + a)(x + a_1)\cdots(x + a_r) \tag{3}$$

BCH 的译码过程分为 3 个步骤,分别是接收码的伴随式的计算、求解错误位置多项式和 Chien 搜索,它们分别由相应的运算电路实现。

3 BCH 编译码器总体设计

3.1 BCH 编译码器参数设计

基于 NAND Flash 存储芯片的特性,文中采用 512 Byte 作为计算单元,设计了一种可对 512 Byte 数据进行 8 bit 纠错的 BCH 编解码器,构造的 BCH 码为(4 200,4 096,8),是本原 BCH 码(8 191,8 087,8)的缩码,并行度 $r=16$,可满足不同页容量 NAND Flash 和应用环境对于 ECC 校验的要求。

文中设计选取的有限域为 GF(2^{13}),GF(2^{13})的本原多项式为 $p(x) = 1 + x + x^3 + x^4 + x^{13}$。根据上述参数设置及式(3),计算出生成多项式,如式(4):

$$\begin{aligned} g(x) = 1 &+ x + x^5 + x^8 + x^9 + x^{11} + x^{12} + x^{13} + x^{14} + \\ &x^{15} + x^{18} + x^{22} + x^{23} + x^{26} + x^{30} + x^{31} + x^{32} + \\ &x^{38} + x^{40} + x^{41} + x^{42} + x^{47} + x^{48} + x^{49} + x^{52} + \\ &x^{58} + x^{59} + x^{64} + x^{65} + x^{67} + x^{68} + x^{69} + x^{70} + \\ &x^{77} + x^{78} + x^{82} + x^{84} + x^{88} + x^{91} + x^{92} + x^{93} + \\ &x^{94} + x^{95} + x^{96} + x^{98} + x^{100} + x^{104} \end{aligned} \tag{4}$$

3.2 编译码器总体架构设计

文中设计的 BCH 编译码器总体架构如图 1 所示,组成模块包括:16 路并行编码器、伴随式计算电路、简化的无求逆 BM 算法电路、Chien 搜索电路、纠错信息 Fifo 和顶层 IO 接口。对外接口包括 16 bit 并行数据、控制信号和纠错信息,编译码器的工作频率为 100 MHz。

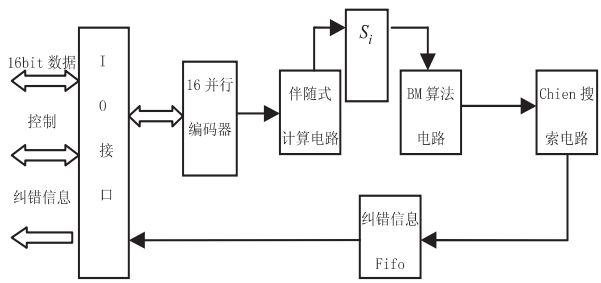

图 1 BCH 编译码器总体架构

大部分 BCH 编解码器的编码电路和译码电路是分别独立工作的,在编码的同时可以进行译码,适合随机读写频繁的应用情况。空间应用中大部分工况为连续读或写操作,在译码时编码电路不工作,并且数据传输速率较大。文中的设计将编码电路复用为译码电路的一部分,并对伴随式计算算法进行相应的改进,可以在基本不增加译码时间的情况下减少大量资源消耗(具体算法设计见 4.2)。

文中的设计采用流水线操作提高数据传输速率:第一级流水为伴随式计算电路(含复用的编码电路);第二级流水为错误多项式求解电路和 Chien 搜索电路。为适应流水线操作,设计一个 Fifo 用于储存纠错信息,可存储不小于 3 个 512 Byte 数据块的纠错信息,纠错信息的数据结构如下:

错误信息:{zero[19 bit],错误数量[4 bit],启动纠错[1 bit],错误类型[2 bit],标志位[1 bit]}

纠错信息:{错误位置地址偏移[10 bit],字掩码[16 bit],标志位[1 bit]}

错误位置地址偏移表示自接收码最高位的字地址偏移量,结合字掩码,确定错误位置。编译码器的错误

类型根据错误的数量分为三种:01—0 错误;10—错误数不大于 8,可纠错;11—错误数大于 8,不可纠错。

4　BCH 编译码器算法设计及 FPGA 实现

4.1　并行编码电路设计

由式(1)和式(2)可知,BCH 编码运算即为对生成多项式 $g(x)$ 进行除余,采用穿行移位寄存器实现的编码运算难以满足速度的要求。文献[9]中提出了一种并行编码算法,可以大幅提升数据吞吐率,并行度为 r 编码电路,编码时间为 k/r 个时钟周期。

假设 BCH 码构造为 (n,k,t),其信息多项式为 $m(x)=m_{k-1}x^{k-1}+m_{k-2}x^{k-2}+\cdots+m_1x+m_0$,将位移运算转变成矩阵计算的形式,实现 r 路并行编码运算:信息多项式表示为向量 $\boldsymbol{I}(t)$,寄存器状态表示为向量 $\boldsymbol{F}(t)$,一次移位运算表示为矩阵 $\boldsymbol{T}_g$,运算公式为 $\boldsymbol{F}(t+1)=\boldsymbol{T}_g^{\,r}\times[\boldsymbol{F}(t)+\boldsymbol{I}(t)]$,详细可见式(5)。其中 g_i 为生成多项式的系数。在文献[9]中给出了算法的详细推导过程,$\boldsymbol{T}_g^{\,r}$ 也被称为 lookahead[9] 矩阵,通过该算法可将 r 次移位运算压缩成一次矩阵乘法运算,使编码电路的速度提升 r 倍。

$$\boldsymbol{F}(t+1)=\boldsymbol{T}_g^{\,r}\times[\boldsymbol{F}(t)+I(t)]=$$
$$\begin{bmatrix}0&0&\cdots&0&1\\1&0&\cdots&0&g_1\\\vdots&\vdots&\cdots&\vdots&\vdots\\0&0&\cdots&0&g_{n-k-r}\\0&1&\cdots&0&g_{n-k-(r-1)}\\\vdots&\vdots&\cdots&\vdots&\vdots\\0&0&\cdots&0&g_{n-k-2}\\0&0&\cdots&1&g_{n-k-1}\end{bmatrix}^r\times$$
$$\left[\begin{bmatrix}F_0(t)\\F_1(t)\\\vdots\\F_{n-k-r}(t)\\F_{n-k-(r-1)}(t)\\\vdots\\F_{n-k-2}(t)\\F_{n-k-1}(t)\end{bmatrix}+\begin{bmatrix}0\\0\\\vdots\\m_{k-(t+1)r}\\m_{k-(t+1)r+1}\\\vdots\\m_{k-(t+1)r+r-2}\\m_{k-(t+1)r+r-1}\end{bmatrix}\right]\tag{5}$$

并行编码电路由移位寄存器和矩阵运算电路组成。文中的设计采用的是 16 位并行计算,要完成对 512 Byte 数据的 BCH 编码,共需要 256 个时钟周期,相对于串行编码的 4 092 个周期,并行编码可以大幅度提高 NAND Flash 芯片的数据吞吐率。

4.2　伴随式计算电路设计

BCH 译码的第一步是要计算接收码 $v(x)$ 的伴随式,可纠错数为 t 的 BCH 码,需要计算 $2t$ 个伴随式,文献[1-2]中所采用的伴随式计算公式如式(6),S_i 计算电路如图 2。如果并行度为 r,图 2 中电路的复杂度即为 r 个常数 FFM(一个乘数为常数另一个为任意数的有限域乘法器)和 1 个 FFA(有限域加法器)[1],计算周期为 n/r 个时钟。也可采用提高电路工作频率的方式减少并行度,由电路的综合结果可知,其最大工作频率为 213 MHz,如数据输入的频率为 100 MHz,电路的并行度最小为 $r/2$ 才能保证译码速度不变。

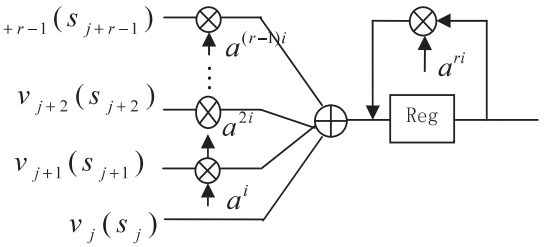

图 2　伴随式求解电路结构图

$$S_i=\sum_{j=0}^{n-1}v_j(a^i)^j=v_0(a^i)^0+v_i(a^i)^1+\cdots+v_{n-1}(a^i)^{n-1}\quad 1\leqslant i\leqslant 2t\tag{6}$$

这种算法需要大量的常数 FFM 和 FFA,资源占用量较大,无法满足航天应用的需求,因此文中提出一种改进算法,可有效地减小资源占用量,具体方法如下:

BCH 码是一种循环码,可以按照循环码的计算方法求解伴随式,该方法分两步:第一步,计算伴随多项式 $s(x)$,计算公式为式(7);第二步,将 a^i 代入多项式 $s(x)$ 求得 S_i[6],计算公式为式(8)。由式(7)可知,伴随多项式的计算也是通过对生成多项式做除余运算来实现的,计算方法与编码过程相同,但计算电路略有不同。为了避免增加电路的复杂度,完全利用原编码电路来实现伴随多项式的求解运算,需要做如下变换:将接收码 $v(x)$ 与 x^{n-k} 相乘,即表示将 $v(x)$ 向高位移 $n-k$ 位,低补零。乘积代入式(7),得到与式(1)相同的表达式,因此,利用编码电路便可直接求得移位后的接收码 $x^{n-k}v(x)$ 的伴随多项式 $s(x)$,多项式次数为 $n-k$,计算周期为 n/r。随后两个步骤的译码运算也就变成了对码 $x^{n-k}v(x)$ 的译码,通过 Chien 搜索得到的错误位置也会较原码的错误位置向低位偏移 $n-k$ 位,例如求得的错误位置为第 x 位,则表示原码的 $x+n-k$ 位错误。因此,最后需要在 Chien 搜索电路中纠正错误位置的偏移。

$$s(x)=v(x)\bmod g(x)\tag{7}$$

$$S_i=s(a^i)=\sum_{j=0}^{n-k}s_j(a^i)^j\quad 1\leqslant i\leqslant 2t\tag{8}$$

由式(8)知,改进算法可以使用图 2 中的电路来实现利用伴随多项式求解 S_i 的计算,与原算法不同的是输入多项式的次数由 n 变为 $n-k$,当求解电路的并

行度为 r 时,此步所需的计算时间仅为 $(n-k)/r$,只占整个计算伴随式的时间的很小一部分。所以,如果降低改进算法中第二步求解 S_i 的电路并行度,因此而带来的时间延时的增加对整个译码速度影响极小,但增大电路并行度却会大大增加译码器的资源占用量。因此文中的设计将伴随式求解电路的并行度设置为 $r/4$,与原算法相比,可以减少大量的资源占用量。

BCH 码的伴随式具有性质[10]:$S_{2i} = S_i^{\ 2}$。可以将改进算法的第二步拆分为两个步骤:先使用图 2 中的电路计算 i 为奇数的 S_i;再使用通用 FFM 计算 S_{2i}。由于对一些 S_{2i} 的计算需要串行进行,如串行计算 S_2、S_4、S_8、S_{16},这样便可以在计算中复用一些通用 FFM,从而进一步减少资源占用量。

以 16 路并行数据输入的(4 200,4 096,8)BCH 编译码器为例,对改进算法和原算法的速度和资源占用量分析对比(电路工作频率均为 2 倍数据输入频率):

(1)原算法的计算周期为 263 个时钟周期;每个 S_i 的计算需要 8 个常数 FFM 和 1 个 FFA,共需要 128 个常数 FFM 和 16 个 FFA。

(2)改进算法的第一步需要 263 个时钟周期,第二步需要 15 个时钟周期,共 278 个时钟周期;改进算法对每个 S_i 的计算需要 4 个常数 FFM 和 1 个 FFA,则对 8 个 i 为奇数的 S_i 的计算共需要 32 个常数 FFM 和 8 个 FFA;对 S_{2i} 的计算共需要 4 个通用 FFM,即大约 28 个常数 FFM 的资源占用量;因此,改进算法共需要大约 60 个常数 FFM 和 8 个 FFA 的资源占用量。

由此可见,与原算法相比,改进算法虽然在计算时间上略有增加,但在硬件复杂度上优势却十分明显。

4.3 错误位置多项式求解电路设计

错误位置多项式求解电路是 BCH 译码电路中最复杂的模块,由于需要大量的通用 FFM,它占用资源也最多,计算也最为复杂。现在主流的错误位置多项式求解算法为 BM 算法,它通过迭代的方法求解错误位置多项式,可纠错数为 t 的 BM 算法需要进行 $2t$ 次的迭代运算,所以其计算时间是三个译码步骤中最短的。文中采用简化的无求逆 BM 算法,主要计算公式如下:使用式(9)求解第 $\mu+2$ 次迭代的差异值;当第 μ 次迭代的差异值不为 0 时,使用式(10)求解第 $\mu+2$ 次迭代的错误多项式 $\sigma^{(\mu+2)}$,ρ 表示第 μ 次迭代前最后一个差异值非 0 的迭代次数。

$$d_{\mu+2} = \sum_{i=0}^{L[\mu+2]} S_{\mu+3-i}\sigma_i^{(\mu+2)} \tag{9}$$

$$\begin{aligned}\sigma^{(\mu+2)}(x) &= d_\rho\sigma^{(\mu)} + d_\mu x^{\mu-\rho}\sigma^{(\mu)}(x)\\ &= d_\rho\sigma^{(\mu)} + d_\mu x^2 B^{(\mu-2)}(x)\end{aligned} \tag{10}$$

算法计算步骤如下,迭代从第 0 次开始:

(1)初始化 $d_\rho = 1$,$d_\mu = S_1$,$\mu = 0$,$B^{(0)}(x) = 2$,$\sigma^{(0)} = 1$。

(2)如果 $\mu = 0$ 则先判断 d_μ 是否为 0,后计算 $\mu = \mu + 2$;否则先计算 $\mu = \mu + 2$,再判断 d_μ。如果 $d_\mu = 0$ 则跳到第 3 步,否则跳到第 4 步。

(3)计算 $B^{(\mu)}(x) = x^2 B^{(\mu-2)}(x)$,$\sigma^{(\mu)}(x) = \sigma^{(\mu-2)}(x)$。

(4)先计算式(10)得到下一步迭代的错误多项式,再计算 $d_\rho = d_\mu$,$B^{(\mu)}(x) = \sigma^{(\mu)}(x)$。

(5)首先判断 μ 是否等于 $2t-2$,如相等,终止迭代,此时错误多项式寄存器 $\sigma^{(\mu)}(x)$ 中的值即为最后结果;如不相等,计算(9)式更新 $d_{\mu+2}$,并回到第 2 步。

算法的电路结构图如图 3 所示,S_i,B_i,σ_i,d_ρ,d_μ 分别代表算法中相应的寄存器。其中寄存器 S_0 应初始化为伴随式 S_1 的值,迭代开始后,伴随式 $S_{2\mu+1}$ 应输入到寄存器 S_0,伴随式 $S_{2\mu}$ 应输入到寄存器 S_1。

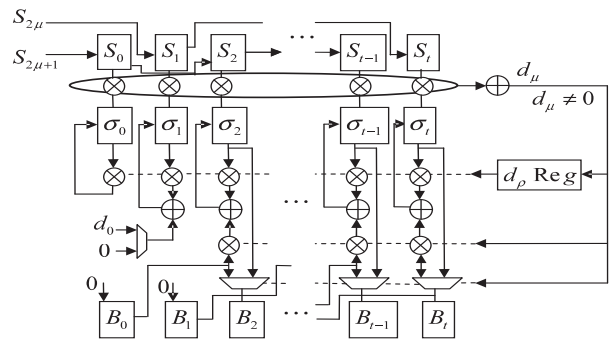

图 3 简化无逆 BM 算法电路结构图

由于式(9)和式(10)的计算都需要大量的通用 FFM 和 FFA,根据文献[11-12]对 FFM 的研究,在 FPGA 上实现的 FFM 电路占用资源较多且延时较大,因此需要将一次迭代运算分配到几个时钟上来完成。而且采用这种方法还可复用不同迭代步骤中的通用 FFM,降低资源占用量。基于上述考虑,并根据对算法电路综合结果的分析,选定 BM 算法电路的工作频率为 2 倍译码器工作频率,一次迭代运算分为 4 拍完成,即一次迭代需要 2 个时钟周期,完成算法总共需要 $2t$ 个时钟周期。

4.4 Chien 搜索电路设计

Chien 搜索算法如下:得到错误多项式 $\sigma(x)$ 后,将有限域 GF(2^m)中所有的非零元素依次代入 $\sigma(x)$,验证 $\sigma(x)$ 是否为零;如果元素 α^i 为 $\sigma(x)$ 的根,则它们的逆 α^{n-i} 的幂便为错误位置,例如元素 α^i 为根,则错误位置为 $n-i$[6]。为提高译码速度,文中采用并行的 Chien 搜索电路,如图 4 所示。图中电路的关键路径为(Tmux + Tm + Ta),它们分别为 MUX、常数 FFM 和 FFA 的关键路径[3],路径延时较小,在布局布线后得到的最大工作频率为 213.8 MHz。综合考虑并

行度与资源之间的反比关系,文中决定采用 8 路并行的 Chien 搜索电路,工作频率为 2 倍时钟频率。

由于选定的 BCH 码为本原码的缩码,码长缩短 3 992 位,减去译码之前位移量 $n-k=104$ 位,码长总共缩短 3 888 位。所以在寻找错误位置时应从第 3 888 位开始向高位搜索,即在进行 Chien 搜索时将 α^i 变为 α^{i+3888}。此运算可通过将第 j 个寄存器 D 中的值初始化为 $\alpha^{(3888\times j)\%8191}\sigma_j$ ($0 \leqslant j \leqslant t$)的方式来实现,并且在搜索到 8088 位时停止搜索。根据上述,计算出所采用的 Chien 搜索电路的计算时间为 263 个时钟周期。

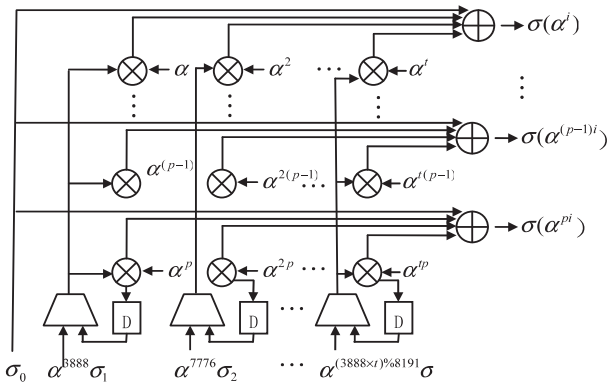

图 4　Chien 搜索电路

5　功能仿真及性能分析

为了验证编译码器的功能,使用一组 512 Byte 的随机数据作为测试用例,经过闭环自测,证明编译码器可以纠正 1 ~ 8 bit 的错误,并返回正确的纠错信息,满足功能要求。为了方便演示,向接收码的最高 8 位注入错误,仿真结果如图 5 所示:由上至下,第 2、3 个信号分别表示错误位置偏移和双字掩码,图中显示错误位置为 262,掩码为 FFOO,由于整个接收码长为 263 个双字,因此可以确定错误位置为第一个双字的最高 8 位;第 5、6 个信号分别表示错误类型和数量,错误数量为 8,错误类型为 2,即表示可纠错。

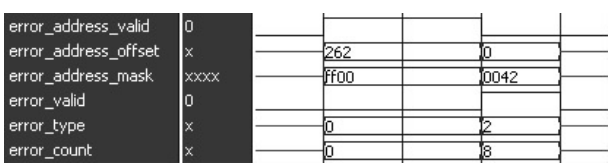

图 5　纠错功能仿真图

对单次译码的速度进行仿真,测得 512 Byte 数据的译码时间为 557 个时钟周期,较原算法增加了第 2 步计算伴随式的 15 个时钟周期。采用流水线操作后,译码时间则为时间最长的第二级流水的时间,即错误位置多项式的计算时间加 Chien 搜索的时间--279 个时钟周期,与原算法的译码时间相同。将设计在 FPGA 上进行综合,分析其实际工作频率和资源占用情况,文中所采用的芯片为 Xilinx 公司的 xc6vsx315t。综合后得到电路的最大工作频率为 128.978 MHz,编译码器的实际工作频率为 100 MHz,编码速率为 200 MB/s,译码速度为 184 MB/s,满足 Flash 芯片的读写操作对于编译码器的速度要求。综合后得到资源占用量为 1 161 个 Slice,与改进前的算法相比节省资源大约 105 个 Slice,资源节省的效果显著。

6　结束语

文中在分析 BCH 编译码原理的基础上,提出了一种适用于航天领域固态存储器 ECC 校验的新型 BCH 编译码器,针对航天应用的特殊需求对译码算法和译码电路架构进行了改进。通过仿真验证及性能分析,证明了该算法在同等的译码速度下,可有效降低译码器对硬件资源的使用量,具有很好的适应性和实用性。

参考文献:

[1] Liu Wei, Rho J, Sung Wonyong. Low-power High-throughput BCH error correction VLSI design for multi-level cell NAND Flash memories[C]//Proc of IEEE workshop on signal processing systems. [s. l.]:[s. n.],2006.

[2] 江建国. BCH 编译码器的设计及验证[D]. 上海:上海交通大学,2010.

[3] Chen Yanni, Parhi K K. Small area parallel chien search architectures for long BCH codes[J]. IEEE transactions on very large scale integration systems,2004,12(5):544-549.

[4] 王　杰. NAND Flash 控制器的 BCH 编/译码器的研究和 ASIC 实现[D]. 杭州:浙江大学,2010.

[5] 王育民,李　晖,梁传甲. 信息论与编码理论[M]. 北京:高等教育出版社,2005:241-249.

[6] Lin S, Costello D J. Error control coding[M]. 2nd ed. New York:Perntice Hall,2004:194-233.

[7] 陈晓东. 分圆陪集的性质及一类 BCH 码的维数[D]. 大连:辽宁师范大学,2008.

[8] 李　超. 本源 BCH 码的维数与周期分布[J]. 国防科技大学学报,1998,20(3):113-117.

[9] Shieh M, Sheu M, Chen C, et al. A systematic approach for parallel CRC computations[J]. Journal of information science and engineering,2001,17:445-461.

[10] Sun F, Devarajan S, Rose K, et al. Design of on-chip error correction systems for multilevel NOR and NAND Flash Memories[J]. IET circuits devices & systems,2007,1(3):241-249.

[11] Reyhani-Masoleh A, Hasan M A. Low complexity bit parallel architectures for polynomial basis multiplication over $GF(2^m)$ [J]. IEEE transactions on computers,2004,53(8):945-959.

[12] 沈晓强. 有限域乘除法研究与实现[D]. 长沙:国防科学技术大学,2006.

BCH编译码器在NAND Flash控制器中的应用研究

作者： 郭鹏， 房亮， 于沛玲， GUO Peng， FANG Liang， YU Pei-ling

作者单位： 郭鹏,GUO Peng(中国科学院 光电研究院，北京 100094；中国科学院大学，北京 100190；中国科学院 空间应用工程与技术中心，北京100094)， 房亮,于沛玲,FANG Liang,YU Pei-ling(中国科学院 空间应用工程与技术中心,北京,100094)

刊名： 计算机技术与发展

ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2014(1)