(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101729077 B
(45)授权公告日 2014.03.12

(21)申请号 200910205192.9
(22)申请日 2009.10.16
(30)优先权数据
08305690.3 2008.10.16 EP
(73)专利权人 汤姆森特许公司
地址 法国布洛涅－比扬库尔
(72)发明人 奥利弗·泰斯 陈晓明 马科·乔治
(74)专利代理机构 北京市柳沈律师事务所 11105
代理人 吕晓章
(51)Int.Cl.
H03M 13/11(2006.01)
G11B 20/18(2006.01)
(56)对比文件
US 5351246 A,1994.12.27, 全文.
CN 101199123 A,2008.06.11, 全文.
US 5644695 A,1997.07.01, 全文.
US 5271012 A,1993.12.14, 全文.
审查员 金霞

权利要求书1页 说明书5页

(54)发明名称

用于二进制数据的错误纠正和错误检测的方法

(57)摘要

本发明提供了一种用于二进制数据的错误纠正和错误检测的方法。为了代数单个符号错误纠正和检测，提出了一种方法，其实现了纠正码字内的未知位置处的单个符号错误；识别码字内的多个符号被不可纠正地毁坏的情况；以及识别码字内的单个符号被不可纠正地毁坏的情况。该方法包括步骤：计算所接收的字的伴随；将该伴随分割为两部分；检查从这两个伴随部分计算的3个整数权重量；将该伴随转换为与所接收的比特相关联的整数值“正交比特错误权重”矢量；以及切换所接收的字的其中关联的“正交比特错误权重”在其可能值范围的上半部分中的那些比特。

CN 101729077 B

1. 一种用于二进制数据的错误纠正和错误检测的方法，所述二进制数据已经由 LDPC 代码进行了错误纠正编码，所述 LDPC 代码的奇偶校验矩阵等于以下 (a) 到 (d) 步骤的结果：

(a) 生成第一中间矩阵，其包括两行方形的、尺寸相同的二进制子矩阵，其中第一行包括 p 个具有尺寸 p•p 的单位矩阵，第二行包括 p 个具有尺寸 p•p 的循环移位矩阵的增幂；

(b) 通过从第一中间矩阵的每个子矩阵中去除在列指数 [r+2ri+i+q] 模 p 处的 m 个等距列，来从第一中间矩阵生成第二中间矩阵，其中 i、m、p、q、r 为整数，m ＞ 1，其中 i ＝ 0，…m-1，其中预定义 m、p、q、r 以使 p ＝ m+2mr，并且其中子矩阵内的列指数从 0 开始；

(c) 通过从第二中间矩阵 H2 的第一行子矩阵中删除只包含零的那些矩阵行，来从第二中间矩阵生成第三中间矩阵；

(d) 向第三中间矩阵预设具有那些行范围的中间行中的“1”元素的、具有高度 2p-m 的 m-1 个二进制列矢量，在所述那些行范围中与移位矩阵的 1 次幂并置的移位矩阵的 0 次幂具有行权重 2；

以字组织该二进制数据，字包括符号，该方法具有步骤：

从所接收的字和 LDPC 代码的奇偶校验矩阵来计算二进制伴随矢量；

将伴随矢量分割为第一子矢量和第二子矢量，所述第一子矢量包括与奇偶校验矩阵的前 p-m 行对应的伴随矢量的 p-m 个比特，所述第二子矢量包括与奇偶校验矩阵的后 p 行对应的伴随矢量的 p 个比特；

通过求和第一子矢量中的比特来从第一子矢量计算第一错误权重，通过求和第二子矢量中的比特来从第二子矢量计算第二错误权重，以及通过求和第二子矢量中除了指数是 2r+1 的整数倍数的元素的比特来从第二子矢量计算第三错误权重；

将第一错误权重和第二错误权重进行比较，并且仅仅如果第一错误权重等于第二错误权重，才继续；

将伴随矢量转换为正交比特错误权重矢量；

经由多数决定，从正交比特错误权重矢量导出多数错误矢量；

从多数错误矢量计算与所接收的字的符号相关联的符号错误权重矢量；

从符号错误权重矢量导出潜在符号错误的数目；

通过接收的字的错误符号与第一子矢量的逐位 XOR 运算，来纠正其中潜在符号错误的数目被导出为 1 的那些所接收的字。

用于二进制数据的错误纠正和错误检测的方法

技术领域

[0001] 本发明涉及用于光存储系统的错误纠正代码(ECC)的领域。其可以被应用于磁记录存储设备、独立磁盘冗余阵列(RAID)系统和传输系统。

背景技术

[0002] H.Fujita 等 人 的“Modified low-density MDS array codes for toleratingdouble disk failures in disk array”,IEEE Trans COMP-56,pp.563-566 呈现了用于容忍磁盘阵列中的双磁盘失效(failure)的低密度 MDS 阵列代码的新的类别。所提出的 MDS 阵列代码具有比 Blaum 等人的 EVENODD 代码更低的编码和解码复杂度。

发明内容

[0003] 一种可高效地编码的准循环(quasi-cyclic)错误纠正代码的方法将在下面被命名为“z 阵列(zArray)代码”。z 阵列代码基于已知的“阵列代码”,如在 R.J.G.Smith,“Easily Decodable Efficient Self-Orthogonal Block Codes”,Electronics Letters,Vol 13 No.7,pp173-174,1977 中发表。z 阵列代码以系统的方式构成具有 LDPC 即低密度奇偶校验类型的 ECC 代码,所述 ECC 代码一方面即使在大的码字(codeword)长度下仍可高效地编码,而另一方面具有在使用消息通过算法(message passing algorithm)被解码时的好的性能。

[0004] 通过下列步骤定义和生成 z 阵列代码的奇偶校验矩阵:生成第一中间矩阵 H1,以包括两行方形的、尺寸相同的二进制子矩阵,其中第一行包括 p 个具有尺寸 p•p 的单位矩阵 I,而第二行包括 p 个具有尺寸 p•p 的循环移位矩阵(cyclic shift matrix)σ 的增幂 σ^u,其中 u = 0,...,p-1。通过从第一中间矩阵 H1 的每个子矩阵中去除在列指数 [r+2ri+i+q] 模 p 处的 m 个等距列,来从第一中间矩阵 H1 生成第二中间矩阵 H2,其中 i、m、p、q 为整数,其中 i = 0,...m-1,其中预定义 m、p、q、r 以使 p = m+2mr,并且其中子矩阵内的列指数从 0 开始。下面,向对应于 σ^u 的子矩阵施加该列去除的结果将被表示为 $\sigma^{u\prime}$。通过从第二中间矩阵 H2 的第一行子矩阵中删除由于去除步骤而只包含零的那些矩阵行,来从第二中间矩阵 H2 生成第三中间矩阵 H3。作为该删除的结果,第三中间矩阵 H3 的第一行包括 p 个具有尺寸 (p-m)•(p-m) 的小单位矩阵 I。通过预设 m-1 个具有高度 2p-m、具有权重(weight)2 的二进制列矢量来从第三中间矩阵 H3 生成 z 阵列代码的奇偶校验矩阵 H,其中列矢量在其中子矩阵的并置 [$\sigma^0\sigma^1$] 具有行权重 2 的那些行范围中的中间行中具有“1”元素。后述二进制列矢量一同被命名为“z”矩阵,因此有 z 阵列代码的名称。

[0005] 下面,我们用 z 阵列码字的“符号(symbol)”来表示这些 p-m 个比特的元组(tuple):所述 p-m 个比特对应于奇偶校验矩阵 H 的在列去除后包含循环移位子矩阵 σ^u 中的一个的列的那些列。进一步地,我们用“符号 x”或“第 x 符号”来表示元组中对应于列 $\sigma^{(x-1)}$ 的那一个。注意在该命名法中,由于它们的编号,所以对应于奇偶校验矩阵 H 的 z 矩阵部分的码字的 m-1 个最左边的比特通常不被视为符号。

[0006] z 阵列代码相比阵列代码的优点：

[0007] - 通过作为列正则 (regular)，z 阵列代码使得能够具有比列非正则 (irregular) 的阵列代码更好的消息通过解码性能；

[0008] z 阵列代码维持随着码字长度而线性增长的编码时间；

[0009] z 阵列代码允许奇偶比特 (parity bit) 生成（即编码）被分割为可调整数目的独立顺序任务，使能编码过程的并行化 (parallelization)。

[0010] 为了高效编码能力和好的消息通过解码性能而设计 z 阵列代码。但是，只在错误由所接收的码字的低比特可靠性来反映时，消息通过解码是恰当的。这可能不是突发错误（或代表具有已知位置的突发错误的擦除 (erasure)）或粒噪声 (shot noise) 事件的情况。在同一符号内包括多个毁坏的比特的单个符号错误的情况中，消息通过解码很可能不能找到正确的码字，尤其是在错误是由某形式的短错误突发导致的情况下。然后，可以执行用于潜在单个符号错误的代数符号错误解码。

[0011] 已经在 1992 年的 M.Blaum 的“A CODING TECHNIQUE FORRECOVERY AGAINST DOUBLE DISK FAILURE IN DISK ARRAYS”中公开了单个错误纠正代码。也分别参见美国专利 No.5,271,012 或 EP 0 519669。这些代码具有最小距离 3，并且因此可以纠正任何单个符号错误。

[0012] 在美国专利 No.5,644,695 中公开了 Blaum 的解码方法。它依靠广义阵列代码，所述广义阵列代码如在美国专利 No.5,351,246、并且也包括美国专利 No.5,271,012 中呈现。

[0013] 尽管 z 阵列代码的大部分码字具有最小距离 3，但它们中的一些具有最小距离 2，从而利用 z 编码并非所有单个符号错误是可纠正的。

[0014] 关于解码 z 阵列编码数据：

[0015] - 对于随机比特错误来说，可以有利地使用消息通过解码；

[0016] - 至少对于 z 阵列代码的子类来说，可以使用来自 z 阵列编码的一个处理步骤的修改来纠正单擦除和双擦除（在码字内的已知位置处的错误符号）。这利用了该子类的结构；

[0017] 对于短突发错误（即在码字内的未知位置处的单个符号的若干比特被毁坏时）的情形来说，到目前为止缺乏有效的解码（即纠正）方法。

[0018] 来自现有技术，即 US5,271,012/EP0519669、US5,644,695 和 US5,351,246 的解决方案涉及不具有允许并行化编码的特征的不同的代码。

[0019] 本发明提供代数单个符号错误纠正和错误检测方法。术语“代数解码”在错误纠正领域中已知，其指其中与被称作“消息通过”的迭代方法相比、从一些给定的数据“计算”正确数据的解码方法，在所述“消息通过”中，给定的错误数据在该方法中渐进地收敛至正确数据。本发明提出并描述：在 z 阵列编码数据上，从现有技术已知的“多数逻辑解码 (Majority LogicDecoding)”的修改可以有效地被用于下列任务：

[0020] - 纠正码字内（在已知位置处）的单个符号错误；

[0021] - 识别码字内的多个符号被不可纠正地毁坏的情况；

[0022] - 识别码字内的单个符号被不可纠正地毁坏的那些（少数）情况。

[0023] 根据本发明的方法包含下列步骤：

[0024] - 计算所接收的字的伴随 (syndrome)；

[0025] - 将该伴随分割为两部分；

[0026] - 检查从这两个伴随部分计算的 3 个整数权重量；

[0027] - 将该伴随转换为与所接收的比特相关联的整数值“正交比特错误权重”矢量；

[0028] - 切换 (toggle) 所接收的字的其中关联的“正交比特错误权重”在其可能值范围的上半部分中的那些比特。

[0029] 优点：

[0030] - 为了 z 阵列编码数据而补足其他以前发明的解码方法。在一起使用时，这些解码技术涵盖许多（如果不是大多数）在实际中重要的解码情景；

[0031] - 在 z 阵列编码数据上使用本方法时，本方法为 US 5,644,695 的解码方法的有利的替代方案。

[0032] 本发明解决纠正 z 阵列码字内的单个符号错误的问题。提出了一种用于纠正 z 阵列码字内的单个符号错误的方法。该方法使用扩展的多数逻辑解码过程。除此之外，多个符号错误和不可纠正的单个符号错误将被识别并且被标记为不可纠正。

[0033] z 阵列代码具有最小符号距离 $d_{min} = 2$。因此，取决于符号中的错误比特的数目，因为定位符号错误的位置不总是可行的，所以不能保证单个符号错误纠正。因此采取措施 (provision) 以至少识别全部不可纠正的符号错误事件。将示出可以使用 z 阵列的所述设计参数“p”来降低这些事件的概率。此外，将识别大多数多个符号错误。

[0034] 根据本发明的方法的优点为：

[0035] - 在单个 z 阵列代码符号的多个比特已经被毁坏的情况中，根据本发明的扩展多数逻辑解码方法的错误纠正概率远高于软决定 (soft decision) 消息通过解码的错误纠正概率；

[0036] - 与软决定消息通过解码相比，根据本发明的解码方法较不复杂，从而需要较少的处理资源。这是由于其硬决定、非迭代的本质。

[0037] - 与美国专利 No.5,644,695 相比，根据本发明的解码方法知晓不可纠正的符号错误事件。对于具有最小符号距离 3 的阵列代码来说，全部单个符号错误都是可纠正的。上面已经指出了 z 阵列代码相比阵列代码的优点。

[0038] 根据本发明，以字组织的二进制数据的错误纠正和错误检测包括步骤：

[0039] - 从所接收的字 r′ 计算二进制伴随矢量 s；

[0040] - 将伴随矢量 s 分割为第一子矢量 s0 和第二子矢量 s1；

[0041] - 从第一子矢量 s0 计算第一错误权重 ws0，并且从第二子矢量 s1 计算第二错误权重 ws1 和第三错误权重 ws1′ ；

[0042] - 将伴随矢量 s 转换为正交比特错误权重矢量 eow；

[0043] - 经由多数决定，从正交比特错误权重矢量 eow 导出多数错误矢量 emaj；

[0044] - 从多数错误矢量 emaj 计算与所接收的字的符号相关联的符号错误权重矢量 esym；

[0045] - 从符号错误权重矢量 esym 导出潜在符号错误数 nsym；

[0046] - 通过与第一子矢量的 s0 的逐位 (bitwise)XOR 运算，来纠正其中导出了 nsym = 1 的那些所接收的字 r′ 。

具体实施方式

[0047] 在下面的描述中更详细地解释本发明的示范实施例。

[0048] 将具有指数 x = 1,...,p 的 z 阵列码字的符号 (symbol) 定义为包括 z 阵列码字的 p-m 个比特的元组,所述 p-m 个比特为了奇偶校验而分别被 z 阵列代码的奇偶校验矩阵 H 的子矩阵 I 和 $\sigma^{(x-1)\prime}$ 相乘。

[0049] 在许多情况中,z 阵列代码允许通过应用扩展多数逻辑解码策略来纠正单个符号错误。单个符号错误被定义为在符号的 p-m 个比特中的至少 1 个处的毁坏。多个符号错误是不可纠正的。假定在解码之前错误事件的种类是未知的。

[0050] 用于单个符号错误纠正的扩展多数逻辑解码

[0051] 下面,到目前为止被表示为“H”的 z 阵列代码的奇偶校验矩阵将被表示为 H_{mz}。

[0052] - 步骤 1 :在 GF2 中从所接收的矢量 (也被表示为所接收的字)r′ 来计算伴随 $s = r' H_{mz}^{T}$。

[0053] - 步骤 2 :检查该伴随 :如果 s = 0 成立,则相信所接收的矩阵 r′ 等于所发送的码字 v ;中断。

[0054] 注意,将只对于非零伴随的情况执行下列步骤。

[0055] - 步骤 3 :从 s 的对应于 H_{mz} 的前 (upper)p-m 行的那些 p-m 个比特来提取 s_0。从 s 的对应于 H_{mz} 的后 (lower)p 行的 p 个比特来提取 s_1。以使 s = [$s_0 s_1$]。

[0056] - 步骤 4 :通过累加 s_0 和 s_1 中的集合比特 (set bit) 来计算整数值错误权重 w_{s0} 和 w_{s1} :

[0057] $$w_{s0} = \sum_{i=0}^{p-m-1} s_0(i) \qquad w_{s1} = \sum_{i=0}^{p-1} s_1(i)$$

[0058] 通过累计 s1 中的集合比特同时忽略 s1 的那些元素来计算 $w_{s1'}$,所述那些元素的计算在步骤 1 中涉及 H_{mz} 的 z 矩阵部分中的“1”元素 :

[0059] $$w_{s1'} = \sum_{\substack{i=0 \\ i \neq j(2r+1)}}^{p-1} s_1(i)$$

[0060] - 步骤 5 :检查错误权重等式 :如果 $w_{s0} \neq w_{s1}$,则已经检测到非单个符号错误 ;用下列选项中断 :

[0061] 选项 5a :如果 $w_{s0} = w_{s1'} = 0$,这指示在 r′ 中,只有与 H 的 z 矩阵部分相关联的部分是错误的,而 r′ 的信息部分 u′ 和其他奇偶比特是无错误的。在该情况中,v_{par1} 的重建是可能的,但可能对其不感兴趣。

[0062] 选项 5b :否则,已经检测到不可纠正的多个符号错误。

[0063] - 步骤 6 :用常规 (即非 GF2) 矩阵乘法来计算正交比特错误权重矢量 e_{ow} :

[0064] $e_{ow} = sH_{mz}$

[0065] e_{ow} 具有与 r′ 相同的维度 (dimensionality)。由于 H_{mz} 的列权重和非 GF2 乘法,所以 e_{ow} 的元素∈ {0,1,2}。

[0066] - 步骤 7 :假设最大正交错误数为 J,将正交比特错误权重矢量 e_{ow} 的分量多数解码为多数错误矩阵 e_{maj}。(根据 Lin, Costello 的“Error ControlCoding”的 872 页中的 17.6.1 节来定义正交性)。这意味着,对于 e_{ow} 的每个元素 n 来说,如果 $e_{ow}(n) > [J/2]$,则

1 被解码，否则 0 被解码。因此，对于 J = 2 的 z 阵列代码来说，多数错误矢量可以被写作：

[0067] $e_{maj} = [e_{ow}/2]$

[0068] e_{maj} 的元素∈ {0,1}。

[0069] （步骤 6 和 7 为传统的众所周知的多数逻辑解码步骤，从这些步骤可以解码 $r = r' \oplus e_{maj}$。）

[0070] - 步骤 8：对于每个符号指数 x = 1，...，p，通过计数每个符号内的多数错误矢量的“1”元素来计算 p 符号错误权重：

[0071] $$\mathbf{e}_{\mathrm{sym}}(x) = \sum_{i=(x-1)(p-m)+z+1}^{x(p-m)+z} \mathbf{e}_{\mathrm{maj}}(i)$$

[0072] 这忽略了对应于 H_{mz} 的 z 矩阵部分的那些来自 e_{maj} 的 z 多数错误，因为它们不在每个定义中定义符号。

[0073] - 步骤 9：对于每个符号 x，检查是否 $e_{sym}(x) = w_{s0}$。错误权重 ws0 为能够出现的最大符号错误权重。

[0074] $$\mathbf{e}_{\mathrm{ws0}} = [\mathbf{e}_{\mathrm{sym}} / w_{s_0}], \in \{0, 1\}$$

[0075] $e_{ws0}(x) = 1$ 指示符号指数 x 处的潜在错误。

[0076] （该步骤可能被解释为基于第二符号的多数逻辑解码步骤，但不应与传统的两步多数逻辑解码器混淆。）

[0077] - 步骤 10：计数潜在符号错误的数目：

[0078] $$n_{\mathrm{sym}} = \sum_{x=1}^{p} \mathbf{e}_{\mathrm{ws0}}(x)$$

[0079] - 步骤 11：检查不可纠正的单个或多个符号错误。

[0080] a. 如果 $n_{sym} = 0$，则已经探测到多个符号错误。所接收的矢量 r′ 中的错误不可纠正；中断！

[0081] b. 如果 $n_{sym} > 1$，则已经探测到单个但不可纠正符号错误。所接收的矢量 r′ 中的错误不可纠正；中断！

[0082] 步骤 12：$n_{sym} = 1$。通过将所接收的矢量 r′ 的错误符号与 s_0 进行 XOR，来纠正对应于符号指数 x_{def} 的单个符号错误，以接收 r，对于所述 x_{def} 来说 $e_{ws0}(x_{def}) = 1$ 成立。

[0083] 注意 $s_0 = e_{maj}((x_{def}-1)(p-m)+z+1 : x_{def}(p-m)+z)$ 成立。

[0084] （步骤 8-12 被视为扩展步骤）

[0085] 也可以为了单个符号错误纠正而将根据本发明的扩展多数逻辑解码应用至根据美国专利 No. 5,271,012 的 Blaum 的阵列代码。在那里，因为代码不具有 v_{par1} 部分，所以可以去掉步骤 5。并且，因为阵列代码不遭受不可纠正的单个符号错误，所以步骤 11 的选项 b 中的条件 $n_{sym} > 1$ 将永不成立。

[0086] 换言之，为了代数单个符号错误纠正和检测，提出了一种方法，其实现了纠正码字内的未知位置处的单个符号错误；识别码字内的多个符号被不可纠正地毁坏的情况；以及识别码字内的单个符号被不可纠正地毁坏的情况。该方法包括步骤：计算所接收的字的伴随；将该伴随分割为两部分；检查从这两个伴随部分计算的 3 个整数权重量；将该伴随转换为与所接收的比特相关联的整数值“正交比特错误权重”矢量；以及切换所接收的字的其中关联的“正交比特错误权重”在其可能值范围的上半部分中的那些比特。