(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 101401076 B
(45)授权公告日 2011.05.18

(21)申请号 200580027871.1

(22)申请日 2005.07.22

(30)优先权数据
60/604,353 2004.08.24 US
10/956,463 2004.09.30 US

(85)PCT申请进入国家阶段日
2007.02.15

(86)PCT申请的申请数据
PCT/US2005/026054 2005.07.22

(87)PCT申请的公布数据
WO2006/023196 EN 2006.03.02

(73)专利权人 桑迪士克3D公司
地址 美国加利福尼亚州

(72)发明人 克里斯托弗·S·穆尔
阿德里尔·杰戴 马特·弗鲁因
杨佳 德里克·博施

(74)专利代理机构 北京律盟知识产权代理有限责任公司 11287
代理人 刘国伟

(51)Int.Cl.
G06F 12/00(2006.01)
G06F 13/00(2006.01)

(56)对比文件
US 2004/0098428 A1,2004.05.20,
WO 98/44420 A1,1998.10.08,
WO 02/15010 A1,2002.02.21,

审查员 武晓冬

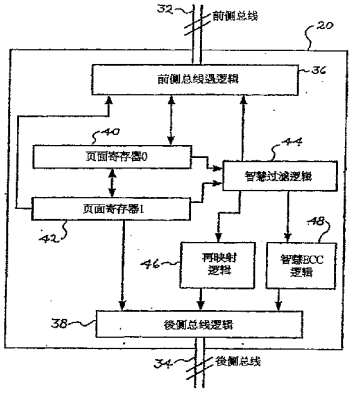
权利要求书 2 页 说明书 17 页 附图 7 页

(54)发明名称

针对具有可擦除/可重写存储器使用带有主机装置的一次或数次可编程存储器的方法及装置

(57)摘要

本文描述的实施例可用于使得一次或数次可编程存储器能与现有消费者电子装置(例如与快闪——可擦除、非易失性存储器一起工作的那些装置)一起工作而不需要一固件升级,从而提供逆向兼容性且同时将用户冲击减到最小。因此,这些实施例是一种将一次或数次可编程存储器与具有闪存卡插槽的现有消费者电子装置桥接的可行方式。这些实施例还允许设计出未来消费者电子装置而不必为了包括针对一次或数次可编程存储器定制的文件系统来更新固件。



CN 101401076 B

1. 一种存储器装置，其包含：

一存储器阵列；和

一控制器，其与所述存储器阵列通信，其中所述控制器可操作以：

(a) 从一与所述存储器装置通信的主机装置处接收一请求以将数据写入到所述存储器阵列中的一第一地址；

(b) 将所述请求写入到所述第一地址的数据与所述第一地址中含有的数据进行比较；和

(c) 基于 (b) 中的所述比较，执行以下操作中的一者：

(c1) 将所述请求写入的数据写入到所述第一地址；

(c2) 在不将所述请求写入的数据写入到所述第一地址的情况下，通知所述主机装置关于所述数据已被写入到所述第一地址；和

(c3) 将所述请求写入的数据写入到一第二地址而并非所述第一地址；

其中所述控制器允许所述存储器装置与使用多次写入文件系统的主机装置逆向兼容。

2. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于含有一组未编程的存储器单元的所述第一地址而执行 (c1)。

3. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以在所述请求写入到所述第一地址的数据附加所述第一地址中含有的数据时执行 (c1)。

4. 根据权利要求 1 所述的存储器装置，其中所述第一地址中含有的所述数据包含一文件系统结构。

5. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于所述主机试图在一文件分配表中写入指示一先前分配的地址是空闲的数据而执行 (c2)。

6. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于所述主机试图将所述请求写入的数据仅写入到一文件列表中的一预定字段而执行 (c2)。

7. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于所述主机试图将所述请求写入的数据仅写入到一最后访问日期字段而执行 (c2)。

8. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于所述主机试图将所述请求写入的数据仅写入到一目录列表中的一预定字段而执行 (c2)。

9. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以在所述请求写入到所述第一地址的数据不同于所述第一地址中含有的所述数据时执行 (c3)。

10. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以在所述请求写入到所述第一地址的数据不同于所述第一地址中含有的所述数据且差异不是仅位于一文件列表中的一预定字段中时执行 (c3)。

11. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以在所述请求写入到所述第一地址的数据不同于所述第一地址中含有的所述数据且差异不是仅位于一目录列表中的一预定字段中时执行 (c3)。

12. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于一格式化所述存储器装置的试图而产生一错误。

13. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于一将所述请求写入的数据写入到一文件分配表的一地址之前的一地址的试图而产生一错误。

14. 根据权利要求 1 所述的存储器装置，其中所述控制器可进一步操作以响应于一将所述请求写入的数据写入到所述存储器阵列中的一预定地址的试图而产生一错误。

15. 根据权利要求 1 所述的存储器装置，其中所述控制器可操作以响应于所述存储器阵列是满的而产生一错误。

16. 根据权利要求 1 所述的存储器装置，其中所述控制器可进一步操作以更新所述存储器阵列中的一文件分配表以指示所述第二地址在执行 (c3) 时已不再是空闲的。

17. 根据权利要求 1 所述的存储器装置，其中将所述存储器阵列组织成可被部分写入的若干逻辑单元，藉此在所述逻辑单元填满之前可对每个逻辑单元执行多个写入操作，且其中所述控制器可操作以在对一逻辑单元进行一最后写入操作时针对所述逻辑单元写入 ECC 数据。

18. 根据权利要求 1 所述的存储器装置，其中所述控制器包含 ：

一第一页面寄存器 ；和

一第二页面寄存器 ；

其中将所述第一地址中含有的所述数据写入到所述第一页面寄存器中，而将待写入到所述第一地址的所述数据写入到所述第二页面寄存器中。

19. 根据权利要求 1 所述的存储器装置，其中所述存储器阵列包含一一次可编程存储器阵列。

20. 根据权利要求 1 所述的存储器装置，其中所述存储器阵列包含一数次可编程存储器阵列。

21. 根据权利要求 1 所述的存储器装置，其中所述存储器阵列包含一三维存储器阵列。

22. 根据权利要求 1 所述的存储器装置，其中所述存储器装置包含一电连接器，所述电连接器经配置以与所述主机装置上的一相应电连接器紧密配合，且其中所述主机装置包含一多次写入文件系统。

23. 一种存储器装置，其包含 ：

一存储器阵列，其存储一文件系统结构 ；和

一控制器，其与所述存储器阵列通信，其中所述控制器可操作以 ：

(a) 从一与所述存储器装置通信的主机装置处接收一请求以在所述文件系统结构中写入指示一先前分配的地址是空闲的数据 ；和

(b) 在不将所述数据写入在所述文件系统结构中的情况下，通知所述主机装置关于所述数据已被写入在所述文件系统结构中 ；

其中所述控制器允许所述存储器装置与使用多次写入文件系统的主机装置逆向兼容。

24. 根据权利要求 23 所述的存储器装置，其中所述文件系统结构包含一文件分配表。

25. 根据权利要求 23 所述的存储器装置，其中所述数据包含 0。

26. 根据权利要求 23 所述的存储器装置，其中所述存储器阵列包含一一次可编程存储器阵列。

27. 根据权利要求 23 所述的存储器装置，其中所述存储器阵列包含一数次可编程存储器阵列。

28. 根据权利要求 23 所述的存储器装置，其中所述存储器阵列包含一三维存储器阵列。

针对具有可擦除 / 可重写存储器使用带有主机装置的一次或数次可编程存储器的方法及装置

[0001] 相关申请案的交叉参考

[0002] 本申请案主张 2004 年 8 月 24 日申请的第 60/604,353 号美国临时申请案的优先权，所述美国临时申请案以引用的方式并入本文中。

技术领域

[0003] 本发明涉及使用一次或数次可编程存储器的方法和装置，其中一次或数次可编程存储器能与现有消费者电子装置一起工作而不需要一固件升级，从而提供逆向兼容性且同时将用户冲击减到最小。

背景技术

[0004] 可以得到模块化、便携式、非易失性存储器装置，所述存储器装置可易于连接到主机装置且与主机装置断开，所述主机装置例如数码相机、数码录音机和个人电脑。这些装置中使用的传统存储器（例如闪存卡）是可重写的，从而使得可出于系统或用户目的而擦除和重写存储器地址。低成本存储器技术（一次可编程的 (OTP) 或数次可编程的 (FTP)）的创造使得主机装置具有新的使用模式，其类似于模拟相机中的胶卷（可简单地照相并印出相片）并同时节约作为“底片”的低成本卡。理想地，能将此低成本卡用在标准闪光相机中，但技术并不一定兼容，因为便携式装置通常使用需要可擦除的系统结构的DOS FAT12/16 文件系统。使用针对 OTP 或 FTP 存储器技术定制的文件系统来更新装置中的固件可以解决这个问题，但这会限制具有软件升级的新型相机的整个可达市场，不然就会要求用户使用所属领域中的软件来升级其现有相机。因此，需要建立一种硬件控制器解决方案，其允许终端用户相对无缝地将 OTP 或 FTP 存储卡用在现有相机中——逆向兼容控制器 (BCC)。

[0005] 现将参考附图来描述优选实施例。

[0006] 发明内容

[0007] 无

附图说明

[0008] 图 1 是优选实施例的主机装置和存储器装置的方块图。

[0009] 图 2 是优选实施例的控制器的方块图。

[0010] 图 3 是优选实施例的存储器图。

[0011] 图 4 是说明优选实施例的智能过滤逻辑操作的流程图。

[0012] 图 5 是优选实施例的 32 字节文件列表结构的图。

[0013] 图 6 是一实施例的 FAT 表的图。

[0014] 图 7 是与图 6 的 FAT 表相关联的文件列表的实例。

[0015] 图 8 和 9 说明优选实施例的智能过滤操作。

[0016] 图 10 是一实施例的用于在边带区域存储重映射信息时读取数据的方法的流程

图。

[0017] 图 11 是优选实施例的动态重映射分配方法的流程图。

[0018] 图 12 是优选实施例的用于重映射实例中的存储器图。

[0019] 图 13 是优选实施例的奇偶检验位布局的说明。

具体实施方式

[0020] 系统概述

[0021] 本文描述的实施例可用于使得一次或数次可编程的存储器能与现有消费者电子装置（例如与快闪——一种可擦除、非易失性存储器一起工作的那些装置）一起工作而不需要固件升级，从而提供逆向兼容性且同时将用户冲击减到最小。因此，这些实施例是一种将一次或数次可编程存储器与具有闪存卡插槽的现有消费者电子装置桥接的可行方式。这些实施例还允许设计出未来消费者电子装置而不必为了包括针对一次或数次可编程存储器定制的文件系统来更新固件。

[0022] 现在转向附图，图 1 是优选实施例的主机装置 5 和存储器装置 10 的方块图。主机装置 5 和存储器装置 10 两者均包含电连接件，所述电连接件彼此配对以将主机装置 5 与存储器装置 10 电耦合。如本文中使用，术语“耦合”意味着通过一个或一个以上介入组件而直接耦合或间接耦合。所述主机装置 5 可采取消费者电子装置的形式，例如（但不限于）数字静态或移动相机、个人数字助理、蜂窝式电话、数字音频播放器或个人计算机（例如具有 USB 读取器 / 写入器或 PCMCIA 卡适配器的那些计算机）。在此实施例中，主机装置 5 含有多次写入文件系统 7，例如 DOS-FAT 文件系统。

[0023] 存储器装置 10 可采用模块化小型手持单元的形式，例如存储卡或棒。存储器装置 10 包含控制器 20 和一次写入记忆体阵列 30。图 2 是控制器 20 的方框图。如图 2 所示，控制器 20 包含分别与前端总线协议逻辑 36 和后端总线协议逻辑 38 耦合的前端总线 32 和后端总线 34。所述前端总线协议逻辑 36 可经配置为多媒体、安全数码卡、存储棒、压缩快闪、智能媒体、xD、USB、HS-MMC 或多种可用的便携式存储媒体中的任一种。所述后端总线协议逻辑 38 可经配置为 NAND 快闪、NOR 快闪或任何其它专有或公共存储器接口。控制器 20 还包含两个页面寄存器 40、42（页面寄存器 0 和页面寄存器 1）、智能过滤逻辑 44、重映射逻辑 46 和智能 ECC 逻辑 48。虽然出于简单起见图 2 中未展示，但存储器装置 10 还可包含计时网络和其它“胶合”逻辑。如下文详细描述，控制器 20 允许存储器装置 10 与使用多次写入文件系统的主机装置逆向兼容。因此，本文中有时将控制器 20 称为“逆向兼容控制器”或“BBC”。

[0024] 控制器 20 的优选实施方案是使用与标准组合逻辑组合的有限状态机 (Mealy 和 / 或 Moore) 的 ASIC。然而，控制器 20 可以各种其它形式来实施，例如（但不限于）具有固件的微控制器或微处理器。此外，尽管在此实施例中控制器 20 通过后端总线 34 而与存储器阵列 30 分离，但可将所述设计集成为单个晶粒以节省成本。在极大程度上，本实施例中的控制器 20 的设计非常类似于用于可重写存储器技术的控制器。差异的例子在于此控制器 20 优选地不需要任何损耗均衡或其它与擦除非易失性存储器相关联的逻辑。尽管此处展现的设计可包括这些额外块和功能，但其可能不是成本最佳的。因此，我们可能希望将它们排除在外。类似地，经设计以与 OTP 或 FTP 存储器一起工作的极低成本控制器可能不需

要一个以上页面寄存器、重映射逻辑或智能过滤逻辑，所述 OTP 或 FTP 存储器使用新的软件文件系统以避免兼容性问题。

[0025] 一次写入存储器阵列 30 包含复数个现场可编程一次写入存储器单元。现场可编程一次写入存储器单元是制作为初始未编程数字状态且可在制作后某时切换到替代性已编程数字状态的存储器单元。举例来说，最初未编程数字状态可被识别为逻辑 1（或逻辑 0）状态，且已编程数字状态可被识别为逻辑 0（或逻辑 1）状态。因为存储器单元是一次写入的，所以存储位置的最初未编程数字状态（例如，逻辑 1 状态）一旦切换到已编程数字状态（例如，逻辑 0 状态）就不能恢复了。代替作为一次可编程（即，一次写入）存储器阵列，存储器阵列 30 可采取数次可编程 (FTP) 存储器阵列的形式，所述 FTP 存储器阵列是可以一次以上但没有多次写入存储器阵列那么多次被写入的存储器阵列。同样，所述存储器装置 10 可含有额外的存储器阵列（一次写入、数次可编程或多次写入）。

[0026] 一次写入存储器阵列 30 可采用任何适合的形式，例如固态存储器装置（即，响应于电子读取和写入信号的记忆体装置）、磁存储装置（例如硬盘驱动器）或光存储装置（例如 CD 或 DVD）。当具有固态存储器单元的形式时，可将存储器阵列 30 中的存储器单元组织成二维或三维形式。在一个优选实施例中，存储器阵列 30 是三维阵列，例如 Johnson 等人的第 6,034,882 号美国专利、Zhang 等人的第 5,835,396 号美国专利和第 09/560,626 号美国专利申请案中所述的阵列，所有这些专利或专利申请案以引用的方式并入本文中。

[0027] 为了说明这些实施例，DOS FAT 文件系统（例如 Windows 操作系统）将用作多次写入文件系统 7 的实例。应注意，这些实施例可与其它类型的多次写入文件系统一起使用。再次转向附图，图 3 是典型的基于 FAT 12/16 的存储卡的存储器图，其展示多个文件系统结构。如本文中使用，“文件系统结构”指的是描述存储器中的分区、所述分区内的存储空间和 / 或文件系统能或不能在所述分区中使用的命令类型的任何数据。文件系统结构的实例包括（但不限于）主启动记录、分区启动记录、文件分配表（“FAT”或“FAT 表”）、根目录和子目录。所述主启动记录 (MBC) 用于存储所述卡（即，存储器装置 10）的启动信息以及每个逻辑分区的开始位置。大多数存储卡（例如此处所示的一者）只含有一个逻辑分区。在此实例中，分区启动记录 (PBR) 含有例如丛集大小、丛集数目、分区类型、分区名称等信息，在地址 0x4000 处开始。FAT1 对应 FAT 表一，所述 FAT 表一含有卡 10 上每个文件的丛集数目链接列表、目录和其它信息。FAT2 对应 FAT 表二。最后，根目录和文件数据区域在物理地址 0xC200 处开始。这些区域被划分成若干丛集，对应于每个 FAT 表中的丛集数目。出于解释本实例目的，假设每个丛集正好是 4KB 大小。注意，这只是 FAT 文件系统的一实例且绝不希望限制本发明。每个结构的不同位置、不同丛集大小和甚至不同文件系统皆可与同此描述的实施例相关联。

[0028] 现描述了系统的一般组件，将论述硬件控制器 20 的智能过滤逻辑 44、重映射逻辑 46 和智能 ECC 逻辑 48，以说明可如何使用这些逻辑块来允许针对可擦除快闪存储器而设计的主机装置几乎无缝地使用 OTP 存储器。这些逻辑块利用这样的事实，即许多现有主机装置不发送擦除命令而是依靠控制器 20 来执行任何擦管理（例如，损耗均衡和垃圾收集）。主机装置仅希望在随后时间处读取一地址时展现写入到所述地址的最近数据。如上文提及，控制器能使用这些逻辑块中的一些或全部，且权利要求书中不需要使用所有三个逻辑块，除非那里明确叙述。

[0029] 智能过滤逻辑 44

[0030] 智能过滤逻辑 44 允许控制器 20 基于所需用户行为而智能地修改主机操作。一种形式的智能过滤逻辑 44 执行先读后写操作以将请求写入到一地址处的数据与先前写入到所述地址的数据进行比较（即，将“新”数据与“旧”数据进行比较）。在最简单的情况下，智能过滤逻辑 44 能在先前写入的数据存在任何修改时执行重映射操作，但这可能导致不良用户模型。通过添加复杂性并智能地查看数据中的差异，控制器 20 可推断主机装置 5 正试图做什么且可接着选择执行以下操作中的一者：将新数据写入到所述地址；“假写入”数据（即，告知主机装置 5 成功地执行了写入操作，而事实上未写入任何内容）；不允许写入并产生卡错误；或执行重映射操作。这在图 4 的流程图中说明。如此图中所示，逆向兼容控制器 20 接收到写入操作（行为 65）。如果数据被写入到空白页面或如果正在附加数据（行为 65），那么控制器 20 根据主机装置 5 请求而写入数据（行为 70）。如果正在更新“最后访问的”字段或试图使 FAT 表“归零”，那么控制器 20 执行“假写入”（行为 80）。如果写入是针对 PBR 的或如果存在重映射带冲突（行为 85），那么控制器 20 产生卡错误（行为 90）。响应于任何其它写入操作（行为 95），控制器 20 执行重映射操作。下文将更详细地描述这些分支。重要的是需要注意到，智能过滤逻辑 44 可执行所述流程图中展示的所有四个分支、少于所有四个分支或未展示的额外分支。

[0031] 如图 4 说明，智能过滤逻辑 44 优选地依据正写入数据的地方来不同地执行。因此，优选的是，智能过滤逻辑 44 不但监视新数据也监视主机所请求的地址。举例来说，适于写入到 FAT 表的过滤可不同于适于写入到用户数据区域的过滤。因此，控制器 20 优选地得知卡 10 上 FAT 表和其它文件系统结构的地址。这可通过将这些地址变成静态并硬编码到逻辑中，通过在启动时让控制器 20 读取并解译 MBR 和 PBR，或通过将地址保存在通电时提供各种配置数据的“魔页”中。

[0032] 为了全面理解智能过滤，首先需要了解用户能对存储器装置执行的操作以及如何将其转译成操作文件系统结构和数据的命令。首先，我们描述在用户照相、复制文件、保存新文件或将文件移动到存储卡上的情况下发生了什么。虽然这些操作可随着不同的装置而有所不同，但其均符合一般趋势。首先，主机装置将读取所述 FAT 表的一者或两者以确定所述卡上哪里有可用的空闲空间。此空闲空间可在 FAT16 文件丛集中表示为 0x0000。接着，主机将把新的 32 字节文件列表的最初 26 字节写入到适当目录丛集中。此 32 字节结构的存储器图可参见图 5。接下来，主机通过写入链接丛集列表而将用于文件的存储器分配到 FAT 表中。在存储器已被分配之后，主机将把文件数据写入到适当丛集。注意，可按任何次序来进行此分配和写入，且也可将其分开以使得每次分配单个丛集和数据写入。在写入了所有数据之后，最后六字节被写入到文件列表中以陈述起始丛集编号和文件大小。

[0033] 在图 6 和 7 中可看到这样的实例。我们假设主机正将 File2.jpg 写入到存储器中的未用空间。首先会将文件名称、属性信息等写入到恰当目录中。接着，会针对所述文件来更新 FAT 表，此处通过将 0x0A 写入到丛集 0x09 中来开始，发信号通知下个丛集在丛集 A 中。直到文件末端被发信号通知已将 0xFF 写入到丛集 0x11 中，才写入链接列表。主机接着将文件数据写入到所述卡的新分配丛集中（再次，可将此操作分成每次分配单个丛集直到到达文件末端为止）。最后，将所述最后六字节写入到文件列表，其展示 File2.jpg 在丛集 0x09 处开始且大小为 45KB。

[0034] 以非常类似于创建新文件的方式来创建新目录。事实上，除了目录属性位之外，目录和文件在 FAT 文件系统实施方案中是完全相同的。因此，主机通过首先将目录名和属性位写入到适当目录来创建目录（在 FAT2 之后的用户数据区域开始处定义 ROOT 目录）。接着，通过将 0x FF 写入到 FAT 表中来分配用于新目录的丛集。任何新建目录内部是分别指向目录和其根源的文件“.”和“..”（“.”文件丛集编号与目录丛集相同）。在写入这两个文件之后，主机接着通过将最后六字节写入到 32 字节列表来完成所述目录列表。

[0035] 用户操作的另一实例是删除先前写入到卡的文件。基于 FAT 的操作系统在用户删除时通常不从卡上擦除文件数据或文件名称。相反地，将字符 0xE5 写入到文件列表的首字节，且通过将 0x00 写入到所述文件的丛集在 FAT 表中的位置来而将其去分配。读取到“E5”作为文件列表中的首字符的任何主机将知道跳过此文件，因为它已经被删除了。通过将正确字符重写到文件中并再分配最初丛集而提供恢复文件的机会。

[0036] 存储器中的每个文件含有 32 字节文件条目，其含有以下信息，例如：文件名称；文件属性，例如只读、隐藏、目录、存档；文件创建日期；文件最后访问日期；文件大小；和第一丛集条目。为了重命名文件或改变其属性（例如，隐藏、存档、只读），主机 5 将仅发送数据写入命令，其关于将新文件名称或属性位写入到最初列表。一般来说，控制器 20 允许主机 5 通过实施重映射操作来修改任何这些参数。然而，控制器可选择通过在更新文件列表中的“最后访问日期字段”（图 5 中字节 18 到 19）时执行假写入来节省存储器。此异常事件的原因在于某些主机（例如 Microsoft Windows）每次读取文件时将把新数据写入到此字段，这会导致使用存储器的 512 字节来进行重映射。这是系统在用户不知道的情况下执行写入操作的实例。为了避免这种情况，智能过滤逻辑 144 可选择不实施此特别更新。将稍后将在本说明书中更详细地论述这些隐藏操作的含意和处理。

[0037] 用户也可选择格式化存储卡。这由主机装置以各种方式来执行。某些装置将仅把 0 写入到 FAT 和根目录，从而“释放”所有存储空间。其它主机将试图通过用新信息重写 MBR 和 / 或 PBR 且接着将主机和根目录归零来重格式化卡。其它主机将执行上述操作中的一项或所有，并且将 0 写入到整个卡以确保所有存储空间被清除。最后，某些主机将执行上述操作中的一项或一项以上，且还使用擦除命令来确保所有数据被完全擦除并为新数据做好准备。

[0038] 用户可尝试的最终操作是非常广泛的——修改现有数据。用户可选择在相机上旋转相片，在使用工具改进相片颜色或另一变量之后将所述相片重保存到卡上，或许多其它操作中的任一者。主机装置可以各种不同方式来执行这些操作中的任一者，所述方式从写入新文件并擦除旧文件到简单地用新数据盖写旧文件的恰当丛集。

[0039] 既然我们基本了解了主机是如何执行不同用户操作的，那么我们可以开始了解智能过滤和它们是如何操作的了。BCC 控制器将在接收到大多数写入命令时执行先读后写 (RBW)。优选地在从主机接收到所需写入地址时立即读取所述存储器装置。这将实现性能改进，因为控制器 20 能读取所需数据并将其插入到页面寄存器 1(PR1)42 中且同时主机 10 正在发送移动到页面寄存器 0(PR0)44 中的新数据。关于此 RBW 操作的更多细节在本文档稍后的重映射描述中。现将结合图 5 展示的 FAT 表来给出智能过滤操作的各种实例。

[0040] 为了创建新文件或目录，主机装置 5 将在恰当子目录中写入新 32 字节文件条目并更新 FAT 表以分配正确的存储空间。如果主机装置 5 正试图写入新文件，其将设法最可能

分别用值 0x13 和 0x14 来分配此图中的最后两个丛集。查看图 4 中的流程图，这将对应于“附加数据”，因为没有任何数据正被修改。因此，所述智能过滤逻辑 44 会简单地将数据写入到最初地址而没有任何改变。在这一般情况下，控制器 20 允许主机装置 5 简单地用新文件盖写空闲空间。然而，在某些情况下，由于先前删除的缘故所述空间可对于主机装置 5 来说看起来是空闲的，但是实际上，存储器重未被擦除。在这种情况下，智能过滤逻辑 44 可实施重映射操作以使得新数据取代旧数据，如下文所述。

[0041] 为了删除文件，主机装置 5 执行两个功能。首先，其用“00h”盖写 FAT 表中的文件的丛集指针，以向主机装置 5 展示所述数据区域现在是可用的。在 OTP 或 FTP 存储器的情况下，很可能存储器不能被擦除，且因此控制器 20 不会想要“释放”存储器那种方式。换句话说，控制器 20 优选地使用重映射算法来使得看起来似乎主机装置 5 具有修改过的数据，但是实际上，永远不能真正擦除并恢复此空闲空间。这就是为什么智能过滤逻辑 44 优选地永远不允许主机将 0x00 写入到先前分配的丛集。此行为在图 4 中被称为“假写入”。具体地说，控制器 20 告知主机装置 5 已将 0 写入到存储器，但实际上不会进行任何写入。重要的是需要了解到，将 0 写入到 FAT 表中会让用户认为在卡上有比实际上更多的空间是可用的，因此造成不良用户经历，即在看起来似乎卡上具有较多空间时，却不能再写入任何新数据。其次，主机装置 5 用“E5h”来盖写 32 字节文件条目的首字符，向 FAT 文件系统表明去跳过所述文件。当主机 5 将写入命令发送给控制器 20 时，控制器 20 执行先读后写操作，以实现先前数据和新数据的比较。这时，智能过滤逻辑 44 启用重映射操作，以用“E5h”来盖写原始文件的首字符，如下文所述。

[0042] FAT 表中的智能过滤的最后实例发生在所述主机 5 试图通过将除 0x00 以外的一些内容写入到先前已写入的丛集来改变数据时。举例来说，如果修改文件且在长度上扩展所述文件（例如，将数据添加到 thumbs.db 文件以使其从 16KB 增长到 24KB），那么可能发生这种情况。此处，会用指向新加数据的丛集编号来取代文件末尾标记 0xFF。在此情况下，会发生重映射操作，从而允许主机 5 根据需要来修改数据。

[0043] 重要的是需要了解到，由于假写入操作，在主机所认为的在存储器中写入的内容与实际上在存储器中写入的内容之间存在偏差。在将 FAT 表高速缓冲存储在易失性 RAM 中的主机系统（例如 Windows）中，这意味着对于单个操作可施加一个以上智能过滤。举例来说，假设用户在一个操作中删除 1MB 文件，且 BCC 20 假造 FAT 表中值 1MB 的丛集的归零。接下来，假设用户试图将 500KB 文件写入到存储器 30。在此情况下，主机 5 将试图把新文件放在由所述删除操作释放的丛集中。因此，其会写入新丛集分配且留下剩余丛集 0x00。智能过滤因此必须重映射新文件丛集，且同时不将先前假造的空间归零。在图 8 和 9 中可以看到这个实例。在时间 0 处，假写入保持卡上的值 11-FF，且在时间 1 处，通过重映射操作而将新文件丛集 11-FF 写入到卡上，而也保持 14-FF 以确保看起来似乎没有释放任何空间。

[0044] 在此实施例中，用于数据区域的智能过滤不同于在 FAT 表中的那些。可存储在用户区域中的数据限于 32 字节文件 / 目录列表和实际文件数据丛集。在最简单的情况下，再次，智能过滤可选择重映射任何正写入在已被使用过的空间上的数据。但是，再次，这会导致不良使用模型，因为早先提及的隐藏系统操作会使用额外存储空间来进行用户并不留心的重映射。为了避免这些多余的重映射操作，BCC 控制器 20 优选地比较页面寄存器 0 与页面寄存器 1 中的数据，且如果唯一的改变或多个改变与特定字段相关联（如图 7 所示），那

么执行假写入。举例来说，如果页面寄存器 0 中的数据与页面寄存器 1 中的数据完全匹配（只是所述两个字节偏移了 18），那么控制器 20 会确定主机 5 曾试图更新“最后访问”字段，且因此会执行假写入操作。因此，主机 5 会认为字段已被更新，但事实上，并未发生任何写入。

[0045] 可将已知文件系统结构字段的任何增量或增量组合应用于智能过滤。举例来说，假定设计者确定 OTP 存储器应不能由用户修改，且因此将不允许任何重命名或删除操作。在此情况下，过滤操作可检测到对于字节 0 到 11 的改变，且可执行假写入。另外，假定设计者决定重命名是可以的，但不允许隐藏文件。如果用户试图重命名文件并改变属性位，那么智能过滤逻辑 44 可检测到这是针对适当字段而发生的且可用新名称来重映射所述数据而并不改变属性位来允许隐藏文件。

[0046] 可由智能过滤逻辑调用的最后操作是发信号通知错误。在优选实施例中，这将仅在两种情形下发生：(1) 如果用户试图格式化卡 10 或 (2) 如果卡 10 已满且不能再写入任何数据。为了格式化卡 10，主机 5 执行至少两个操作：清除 FAT 表且用所需格式信息来更新 PBR。在此情况下，控制器 20 检测到在 FAT 表（图 3 中地址 0x0000-0x4000）上正发生写入操作，且将在响应主机命令时设置某组错误位。这些可以是 CRC 错误、写入失败错误或适合于卡形式因数的任何其它状态错误信号。另外，控制器 20 可以再次“假造”任何试图将 FAT 表归零的写入。当卡 10 已满且不能再写入任何数据时，控制器 20 能从 FAT 表或写入地址中检测到卡 10 已满。在这个时间点上，不能写入任何额外数据，且发信号通知一错误。此外，可设置任何写入保护位来发信号通知用户关于 OTP 卡 10 现在已满且实质上就是 ROM。控制器 20 可以多种方式来检测到卡 10 已满，所述方式从检测 FAT 表到发现重映射寻址与主机写入寻址重叠。

[0047] 先前实例都假设主机不发送实际擦除命令来删除文件或系统数据，而是依赖控制器来执行此操作。然而，实际上，某些主机确实具有额外复杂性，需要所述额外复杂性来由其自身执行擦除操作或使物理和逻辑块失效并在日后执行擦除操作。类似这样的系统实例是智能媒体接口，其中所述说明书展示如何在存储器的边带区域中建立逻辑块，标记无效旗标，且执行其它更复杂的操作。重要的是需要注意到，智能过滤概念也可容易地应用到此类系统，其中控制器会基于旧数据与新数据的比较来智能地执行设置操作，而不管所述操作是写入、擦除还是其它命令。

[0048] 总之，当观看主机可执行的每个个别操作时，智能过滤逻辑可能看起来相对较复杂。然而，通过更仔细观察，可以看到所述行为的多种模式。如果控制器 20 在销售时已被预格式化，那么 PBR、FAT 表、根目录和数据区域的地址都是已知的。如果主机 5 试图盖写 PBR，那么控制器 20 优选地强加卡错误，因为这将改变卡 10 的格式。由于存储器阵列 30 的 OTP/FTP 本质技术的缘故，控制器 20 优选地通过将 00h 写入到 FAT 中先前写入地址来假造任何试图释放卡 10 上的空间的写入。类似地，可假造针对例如最后访问字段的文件条目的不良更新以获得改进的存储器利用。如果主机 5 写入到未曾使用过的存储空间，那么智能过滤逻辑优选地允许不中断地继续进行写入。并且，为了谨慎起见，如果主机 5 以智能过滤逻辑 44 并不理解的方式来盖写先前数据，那么可优选地重映射所述写入以允许成功地继续进行主机操作。最后，优选的是留心可能导致控制器 20 针对单个写入命令来执行多个功能的过滤组合。举例来说，如果主机 5 已高速缓冲存储了所述 FAT 表并试图擦除大文件且接着在

“释放的空间”中复制小文件，那么控制器 20 优选地允许在不释放 FAT 表中的额外空间的情况下写入新文件（经高速缓冲存储的 FAT 与已被过滤的实际 FAT 的不匹配造成这些组合问题）。

[0049] 重映射逻辑 46

[0050] 重映射逻辑 46 可用于将数据移动到一新逻辑地址，所述新逻辑地址允许主机装置 5 认为其已擦除了一特定地址。重映射是存储器控制器 20 用来描述数据和地址间接的术语。其指的是在不同于主机 5 所提供的逻辑地址的物理位置处写入数据的概念。重映射由各种技术（包括快闪存储器）出于多种原因使用，所述原因从性能改进到通过损耗均衡算法来改进数据完整性。然而，对于由 BCC 启用的 OTP 或 FTP 存储器来说，重映射的独特之处在于其用于模仿可擦除性。举例来说，如果主机请求将数据集 A 写入到地址 0，那么控制器 20 可选择将数据写入在地址 0。随后，主机 5 可能想要将数据从 A 改变到 B，例如在写入字符“E5”来执行擦除操作（如早先所述）时。此时，控制器 20 会使用智能过滤来确定主机 5 已恰当地修改数据集 A 并会将数据集 B 写入到不同物理位置（例如地址 1）。现在，当主机 5 读取地址 0 时，控制器 20 必须知道代替地返回位于地址 1 处的数据，因而使得对于主机 5 和用户来说存储器看来已被修改过了。

[0051] 存在许多用于使来自主机装置的数据改道的方法。参见（例如）下列专利文献，所述专利文献转让给本发明的受让人且以引用的方式并入本文中：第 09/877,719 号、第 09/878,138 号、第 10/327,680 号和第 10/023,468 号美国专利申请案。一种常用方法是使用物理到逻辑地址表，在启动时读取或计算所述物理到逻辑地址表且接着将其存储在控制器 20 的易失性存储器中，但这个表可能非常大且会增加控制器 20 的晶粒大小和成本。对于本文所述的实施方案来说，优选的是在不损害性能应用要求的情况下保持尽可能低的成本。在优选实施例中，存在两种重映射方法。

[0052] 第一种也是最简单的方案是使用主机不可见的存储器页面的“边带区域”。在目前优选实施例中，将存储器阵列 30 组织在 528B 页面中，但 FAT 文件系统使用 512B 扇区。这些额外的 16B 被称为边带，且可由控制器 20 使用来存储额外信息，所述信息包括（但不限于）ECC 数据、数据和地址有效性旗标以及重映射指针。在此方案中，如果页面已被重映射，那么控制器 20 使用此边带的一部分来存储物理地址。如果第一次写入所述页面，那么将不需要任何重映射指针，因为物理和逻辑地址是相同的。当主机 5 试图修改数据时，控制器 20 将把新数据写入到新位置且把此位置的物理地址存储到旧位置的边带中。

[0053] 图 10 含有说明在此重映射方案下如何读取数据的流程图。首先，将读取命令和地址发送到卡 10（行为 110）。控制器 20 知道读取整个 528B 页面（或包括指针所需的那么多）（行为 120）且确定边带是否含有 FFh（行为 130），此让控制器 20 得知所述地址是否已被重映射了。如果指针是未曾被用过的（即，含有 FFh），那么控制器 20 将所述 512B 页面返回给主机 5。如果边带含有地址，那么控制器 20 读取存储在边带中的页面地址以找出最近写入的数据。可直接将这个地址存储在边带中或可存储地址的偏移量。这种形式的重映射被称为静态的，因为控制器 20 不修改 FAT 表来分配用于重映射的额外空间。在这个实施方案中，卡 10 已由已经分配在 FAT 表中的一组数量的重映射数据来预格式化。一旦使用了此数量的数据，就不会再发生重映射了。FAT 表展示重映射分配，以使得主机 5 不会出于其自身目的（例如，存储图像数据）而试图使用此空间。

[0054] 这第一种重映射方案是非常强有力的且需要非常少的逻辑额外开销来实施。然而，其确实遇到相对较高的性能额外开销，因为已被重映射 N 次的页面将需要 N 次读取来找出当前数据。另一种简单的重映射方案可被代替使用以改进此性能。此处，不是存储针对经重映射的页面的指针，而是将利用针对重映射表的指针。此表可使用时间到空间映射技术来存储最近的页面数据。时间到空间映射在第 6,738,883 号美国专利中描述，所述专利转让给本发明的受让人并以引用的方式并入本文中。现在，已被重映射 N 次的页面可最多被读入三次，从而显著改进了产品性能且具有非常少的额外成本和复杂性。

[0055] 应注意到，每当访问一地址时，控制器 20 就优选地执行这些读取来找出最新页面。对于早先描述的每个智能过滤操作（其中控制器 20 执行先读后写）来说，控制器 20 也优选地沿着重映射链来找出最近页面。

[0056] 还应了解，可易于向这些算法添加额外复杂性。举例来说，这些算法中的一者或一者以上可与另一者一起使用以在使复杂性减到最小的同时改进性能。另一复杂性实例是使用 FAT 文件系统中的强大数据位置相关性来执行经高速缓冲存储的读取操作，其中控制器 20 会读取一页面或其附近的多个页面来查找数据，并行地正由主机 5 或控制器 20 执行其它操作。尽管这将向控制器 20 增加某些成本，但它可显著减少先读后写和重映射行进额外开销。

[0057] BCC 控制器 20 可实施的与标准重映射算法的另一个较大区别用于解决主机 5 的流量控制问题。换句话说，如果控制器 5 正在主机不知道的情况下将数据移动到新位置，且主机 5 认为此数据实际上正被擦除并重写入，那么主机 5 如何知道实际上有多少可空闲空间呢？为了解决这个问题，BCC 20 实施两个独立操作。首先，如早先所述，智能过滤逻辑 44 不让主机 5 在 FAT 表中写入 0 来释放空间。这样将防止用户控制的操作（例如删除文件）看起来似乎擦除并打开了存储器。然而，也必须解决控制器 20 为重映射而消耗的存储器。

[0058] 解决这个问题的一种简单方式是在销售存储器装置 10 之前为重映射分配存储器。这个静态量的重映射可由控制器 20 使用，直到不剩下任何预分配的空间为止。此时，控制器 20 可发信号通知卡错误，允许不是重映射的任何写入或许多其它操作。可以看到，这种方法（被称为静态重映射分配）可能是不合需要的，因为其限制了用户能执行的操作的数目和大小。

[0059] 一种更复杂形式的重映射（称为动态重映射分配）解决了这个问题。在图 11 中可以看到这种方法的流程图。首先，主机 5 发送写入命令，且将数据存储在页面寄存器 1 中（行为 160）。接下来，控制器 20 执行先读后写 (RBW) 操作并将所述数据存储在页面寄存器 2 中（行为 170）。如果页面寄存器 2 是空白的（即，含有 FFh）（行为 180），那么将页面寄存器 1 的内容写入到存储器阵列（行为 190）。如果不是，那么控制器 20 确定页面寄存器 1 的内容是否等于页面寄存器 2 的内容加上附加数据（行为 200）。如果是，那么将页面寄存器 1 的内容写入到存储器阵列（行为 190）。如果不是，确定智能过滤逻辑是否被启用（行为 210）。如果是，那么过滤数据（行为 220）。如果不是，那么需要重映射（行为 230）。为了执行重映射，找到位于存储器顶部的第一空闲页面（行为 240），且将页面寄存器 1 的内容没有错误地写入到最高的空闲页面（行为 250）。接着，更新 FAT 表来将页面丛集标记为已使用的（行为 260）。最后，将新页面地址写入到最初页面的边带中（行为 270）。尽管此处展示两个页面寄存器来提供性能节约（例如，在地址可用时且在完成来自主机 5 的数据输

入之前发生先读后写），但能够使用单个页面寄存器。

[0060] 如流程图中所说明，控制器 20 通过写入 FAT 表来通知主机 5 空间已被用于重映射。当需要额外存储器时，控制器 20 可分配新丛集以供其自身使用，就像主机 5 会为文件数据分配丛集一样。也就是说，此方案使用相同的指针跟踪算法（如上所述），而且也因为需要额外空间来进行重映射而动态更新 FAT 表。此实施方案的好处在于主机 5 和用户不在其能执行的重映射 / 文件修改数目上受到限制。控制器 20 将允许其重映射，直到卡被完全填满为止。在此优选实施例中，如图 12 所示，控制器 20 从下往上分配存储器，而大多数主机从上往下分配存储器。这种分配方案将允许控制器 20 和主机 5 独立分配存储器而不太可能会发生分配冲突。这还允许控制器 20 每当所述两个分配区冲突时认识到卡 10 已满。当卡已满时，控制器 20 可选择设置其永久和暂时写入保护寄存器，发信号通知主机 5 关于不能再对卡 10 进行任何写入。非高速缓冲存储系统将知道还剩多少存储器，而高速缓冲存储系统将在启动时读取 FAT 表来看在所述特定会话之前使用了多少重映射空间。

[0061] 再次，应注意到静态与动态重映射不是彼此排斥的。事实上，为了辅助用户，可选择在单个产品中实施两者。举例来说，用户可能想要删除或修改文件，即使卡已满且不能添加任何额外文件。如果一组数量的存储器被预分配且在正常操作中不被使用，那么可需要额外数据修改。举例来说，卡 10 可留出 500KB 静态重映射，控制器 20 直到由于主机与控制器分配冲突而使卡 10 呈现满载时才使用所述 500KB 静态重映射。此时，控制器 20 可允许将主机的额外数据写入到静态分配区，直到完成所需操作为止。然后，智能过滤可用于允许某些用户操作（例如删除和重命名）在静态区域中发生，而其它操作会导致错误，因为卡 10 对于大多数使用来说基本上已满。

[0062] 在又一实例中，此静态分配区域可由已用 BCC 技术的软件更新的主机访问。PC 或其它电子装置可由一应用程序启用，所述应用程序向卡发送一信号，所述信号会允许应用程序显式访问卡存储器的全部，如第 6,424,581 号美国专利中描述，所述专利转让给本发明的受让人并以引用的方式并入本文中。此时，所述应用程序会优先于任何智能过滤、重映射等，并可用来通过优化重映射链或甚至允许额外操作进入静态区域中来改进性能，而非启用系统不能做到这些。因此，组合静态与动态重映射算法可使用启用和非启用系统两者来提供最佳用户体验。

[0063] 另外，其它实施方案能够取代上述简单的指针跟踪方案。下文列举一些实施方案：

[0064] 读取 3 命令：可使用较快读取命令（称为 READ3），其会允许控制器 20 只读取边带数据。这会减少在找到最近数据之前读取许多满页面数据的额外开销。

[0065] 指针表：为了减少读取每个页面来找到最近写入的性能损失，可使用指针表，以使得每个页面会具有列举所有重映射的表。主机 5 会读取所述表且使用未曾被用过的条目之前的最后一个指针来确定最近更新。表地址本身会存储在页面的边带中。这类似于上文所参考的专利申请案中所述的硬件重映射中使用的表。

[0066] 系统高速缓冲存储器：控制器 20 可使用 SRAM 或类似高速缓冲存储器来存储来自重映射表的数据或存储在边带中的指针。尽管额外 SRAM 耗费区域且因此使得控制器 20 更昂贵，但这种结构会允许重映射特别快速地发生且使得性能额外开销减到最小。

[0067] 重映射算法组合：控制器 20 可建立重映射利用旗标，所述旗标会表示正在使用简

单的指针跟踪方案还是表方案。因此，如果控制器 20 看到页面已被重映射多次，那么其可选择建立表以减少未来重映射的影响。但是对于简单情况（其中一页面只被重映射两次而已）来说，指针方案就足够了。控制器 20 可动态设置此旗标，或其可为静态的且基于重映射位置的。举例来说，FAT 表将被重映射多次且因此其适于一直使用表或 SRAM 方案，而数据不经常被重映射且因此此处其可优选使用指针跟踪方案。此实施方案可减小 SRAM 成本，且同时仍提供接近最佳的系统性能。

[0068] 智能 ECC 逻辑 48

[0069] 优选的是，使用 ECC 以防止可能会发生在字段中的可靠性错误。在目前优选实施例中，BCC 20 使用边带来存储 ECC 数据以及上述重映射指针。ECC 可以是 8 字节、16 字节或适于边带可用性的任何大小。在目前优选实施例中，使用 128B ECC 算法，本文中将其称为“智能大块 ECC”。大块 ECC（“LB ECC”）对于每个 ECC 字使用两个关闭且在附录 1 中进行论述。根据正常大块 ECC 方案，由于 BCC 20 利用边带来用于重映射和 ECC 信息两者，因而边带可能没有足够的空间来对每个字存储两个 ECC 关闭。相应地，通过仅将每个字关闭一次——当已写入字的最后两个字节时，来使得此大块 ECC 方案成为“智能”。选择两字节的值是因为 FAT 表中的每个条目的大小为 2 字节。（当执行动态重映射连同 LB ECC 时，控制器 20 优选考虑 ECC 关闭操作来确保字段中产品的最大可靠性。换句话说，分配优选不应关闭 FAT 表中的 ECC 字并接着在日后更新相同的字。）还可根据数据区域来进行所述关闭操作。举例来说，当更新 FAT 表时，可能适于基于两字节片段来更新（如上所述），但当更新数据区域时这样做就可能没有意义了。当更新数据区域时，可基于字中的最后六字节来执行关闭操作，因为大多数主机在完全写入所有数据之后仅更新文件条目的最后六字节。最后，如果检测到正在更新先前关闭的 ECC 字，那么 BCC 20 可选择实施重映射操作。这会利用重映射操作结合 ECC 来保持字段中的最大可靠性。

[0070] 第 6,738,883 号美国专利展示使用 ECC 作为改进 OTP 或 FTP 产品中数据可靠性的方法，所述专利转让给本发明的受让人且以引用的方式并入本文中。如此专利中论述，ECC 字可被写入一次且仅一次。盖写 ECC 字中的新数据会导致 ECC 数据值不正确，从而移除了产品所需的可靠性层面。在已被使得能与 OTP 或 FTP 存储器一起工作的系统中，可易于通过为存储器建立线概念（最小写入大小）来处理这个问题，这在上文参考的’883 专利中论述。然而，优选地用设计复杂性来补偿这样的情形，其中 BCC 产品用于非启用系统（即，不具有八字节对准概念的主机）中。

[0071] BCC 制器 20 可以很多方式来维持 ECC 完整性。一个简单方法是每当 ECC 字失效时就调用重映射操作。此处，控制器 20 会检测 ECC 字是否正由智能过滤或其它逻辑方案盖写。在使 ECC 字失效的情况下，重映射操作会用新 ECC 数据来重写所述数据，因而保持非常可靠的产品。

[0072] 尽管重映射可用于保持存储器的 100％ ECC 覆盖，但其会导致许多额外重映射，从而限制了用户的操作数目。在此情况下，可设计控制器 20 以改进 ECC 处理。在优选实施例中，将不写入 ECC 数据且 ECC 字将在直到写入数据的特定部分时才被完成。举例来说，假设 ECC 字是 32 字节。在先前所述的写入文件的情况下，主机 5 将写入 26 字节的文件列表数据且稍后在完成了文件数据之后写入最后六字节。在此情况下，控制器的 ECC 智能过滤逻辑 48 可确定正在写入文件列表，且直到写入最后六字节时才完成所述 ECC 字。这会维持 ECC

完整性，且同时将重映射额外开销减到最小。可针对多种 ECC 字大小、文件系统结构和数据结构来修改此方案。

[0073] 总之，智能ECC逻辑48在目前优选实施例中使用症状位存储在边带中的128B ECC字，基于写入最后两字节数据来关闭字，只允许每个字关闭一次，针对 FAT 表与数据区域使用不同的关闭方案（2B 与 6B 关闭），且在所述 ECC 字失效或破损时可使用重映射。在此实施例中使用智能 ECC 逻辑 48，因为边带空间不足以进行多个 ECC 关闭。

[0074] 结论

[0075] 存在多种可与这些优选实施例一起使用的替代方法。举例来说，代替在存储器装置的控制器中实施，智能过滤逻辑、重映射逻辑和智能 ECC 逻辑中的任一者或所有都可整体或部分地在另一组件中执行，所述另一组件例如存储器装置中或主机装置中的另一控制器。另外，术语“电路”指的是可实施针对其而描述的功能的任何组件。电路的实例包括上文给出的实例以及运行计算机可执行指令的处理器、专用集成电路、现场可编程门阵列、模拟电路或现在已知的或日后开发的任何其它硬件装置（运行或不运行软件）以及其任何组合。

[0076] 希望将前述详细描述理解为本发明可采取的选定形式的说明而不是本发明的定义。希望只有所附权利要求书（包括所有等效物）来界定本发明范围。最后，应注意到，可单独或彼此组合来使用本文描述的优选实施例中任一者的任一方面。

[0077] 附录 I

[0078] 大块 ECC 实施

[0079] 概述

[0080] 大块 ECC 优选地包括以下各项：

[0081] • 128B 的 ECC 字大小。

[0082] • 单错误校正，但无双错误检测。

[0083] • 智能 ECC 更新，其允许最多两个 ECC 奇偶性更新。

[0084] SEC ECC 实施

[0085] ECC 实施类似于允许对每个字节计算 ECC 奇偶校验位的实施。由于侧带中的存储有限而需要减少奇偶校验位，已仅针对 SEC 而不含 DED 进行了下列修改。图 13 展示奇偶校验位针对 8 位数据如何运作的实例：

[0086] • P0 覆盖 D1、D3、D5、D7。

[0087] • P1 覆盖 D2、D3、D6、D7。

[0088] • P2 覆盖 D4、D5、D6、D7。

[0089] • P3 覆盖 D0、D1、D2、D3。

[0090] • P4 覆盖所有数据位。

[0091] 下表展示任何一位错误会导致的症状。症状 [4:0] 是由对期望和实际奇偶校验位进行异或运算造成的。这种算法优选需要 N+2 个奇偶校验位来覆盖 2^N 个位。针对 128 字节（210 位）的 ECC 字，优选需要 12 个奇偶校验位。

[0092]

错误位	症状 [4:0]

D0	11000
D1	11001
D2	11010
D3	11011
D4	10100
D5	10101
D6	10110
D7	10111
P0	00001
P1	00010
P2	00100
P3	01000
P4	10000

[0093] 12 位奇偶等式

[0094] 下列是 1024 位 ECC 字优选所需的 12 组奇偶等式（全部为 1 的 ECC 字应导致全部为 1 的奇偶校验位）：

[0095] • P0 =！ XOR(D1, D3, D5, . . . , D1023)

[0096] • P1 =！ XOR(D2-D3, D6-D7, . . . , D1022-D1023)

[0097] • P2 =！ XOR(D4-D7, D12-D15, . . . , D1020-D1023)

[0098] • P3 =！ XOR(D8-D15, D24-31, . . . , D1016-D1023)

[0099] • P4 =！ XOR(D16-D31, D48-D63, . . . , D1008-D1023)

[0100] • P5 =！ XOR(D32-D63, D96-127, . . . , D992-D1023)

[0101] • P6 =！ XOR(D64-D127, D192-D255, . . . , D960-D1023)

[0102] • P7 =！ XOR(D128-D255, D384-D511, . . . , D896-D1023)

[0103] • P8 =！ XOR(D256-D511, D768-D1023)

[0104] • P9 =！ XOR(D512-D1023)

[0105] • P10 =！ XOR(D0-D511)

[0106] • P11 =！ XOR(D0-D1023)

[0107] 智能 ECC 更新

[0108] 智能 ECC 更新优选地允许针对每个 ECC 字两次写入奇偶校验位。此特征允许部分

页面写入由 ECC 覆盖。在针对 ECC 字第一次写入新数据期间写入第一组奇偶校验位。任何随后新数据写入优选地将使第一组奇偶校验位无效。在页面正被“关闭”时写入第二组奇偶校验位。当 ECC 字的最后八字节中的任一位均被编程时认为页面被“关闭”。这种模型与需要部分页面写入的 FAT 结构合作良好。

[0109] 智能 ECC 旗标和奇偶校验位优选地仅在新奇偶校验位与现存有效奇偶校验位不匹配时被更新。相同数据的再写入（其将导致匹配的奇偶校验位）将不会改变智能 ECC 旗标的状态。

[0110] 未写入的字 :读取时启用 ECC

[0111]

111111111111	11	11	111111111111	11	11
奇偶性 0[11:0]	未使用 0[1:0]	有效 0[1:0]	奇偶 1[11:0]	未使用 1[1:0]	有效 1[1:0]

[0112] 上文展示针对尚未被编程的数据字的 ECC 奇偶性、未使用和有效位。有效和未使用位每一者都被复制两次以获得冗余。在此状态下，由于奇偶性 0 是有效的，因而在读取时启用 ECC。注意到，如果有效 0 是 1，那么不管未使用 0 是什么值。（这对于奇偶性 1 位并不正确。）

[0113] 以上奇偶性、未使用和有效位的数目是 32 位或 4B。每个页面具有四个 128B ECC 字，每个页面存在 16B 的 ECC 奇偶性、未使用和有效位。

[0114] 奇偶性 0 写入 :读取时启用 ECC

[0115]

100101101110	00	11	111111111111	11	11
奇偶性 0[11:0]	未使用 0[1:0]	有效 0[1:0]	奇偶性 1[11:0]	未使用 1[1:0]	有效 1[1:0]

[0116] 上文展示已被编程一次的数据字的 ECC 奇偶性、未使用和有效位。由于未使用 0 先前为 1，我们知道可编程奇偶性 0 来反应数据正被写入。未使用 0 也被编程为 0 以指示奇偶性 0 已被写入。在此状态下，由于奇偶性 0 仍然有效，因而在读取期间启用 ECC。

[0117] 奇偶性 0 失效 :读取时禁用 ECC

[0118]

100101101110	00	00	111111111111	11	11
奇偶性 0[11:0]	未使用 0[1:0]	有效 0[1:0]	奇偶性 1[11:0]	未使用 1[1:0]	有效 1[1:0]

[0119] 上文展示通过将有效 0 编程为 0 而使得其奇偶性 0 位无效的数据字的 ECC 奇偶性、未使用和有效位。在第一次写入数据字之后，如果使用 0 已经被编程为 0，那么随后写入将使得奇偶性 0 位失效。在此状况下，由于奇偶性 0 已经失效且奇偶性 1 有待更新，因而在读取时禁用 ECC。

[0120] 奇偶性 1 更新 :读取时启用 ECC

[0121]

100101101110	00	00	100101101110	00	11
奇偶性 0[11:0]	未使用 0[1:0]	有效 0[1:0]	奇偶性 1[11:0]	未使用 1[1:0]	有效 1[1:0]

[0122] 上文展示使其奇偶性 1 位被写入的数据字的 ECC 奇偶性、未使用和有效位。此发生在所述字的最后八字节中的任一位被编程且未使用 1 先前为 1 时。未使用 1 也被编程为

0 以指示奇偶性 1 已被写入。在大多数情况下，当最后八字节正被编程时，这是对所述数据字的最后部分页面写入。在此状态下，由于奇偶性 1 是有效的且被写入，因而在读取时启用 ECC。（与奇偶性 0 位不同，未使用 1 旗标必须为 0 以使得奇偶校验位有效。）

[0123] 奇偶性 1 失效：读取时禁用 ECC

[0124]

100101101110	00	00	100101101110	00	00
奇偶性 0[11:0]	未使用 0[1:0]	有效 0[1:0]	奇偶性 1[11:0]	未使用 1[1:0]	有效 1[1:0]

[0125] 上文展示通过将有效 1 编程为 0 而使其奇偶性 1 位失效的数据字的 ECC 奇偶性、未使用和有效位。在奇偶性 1 位被写入之后，如果存在针对此数据字的随后写入，那么奇偶性 1 位必定会失效。在此状态下，由于奇偶性 0 和奇偶性 1 都无效，因而在读取时禁用 ECC。

[0126] 解码冗余未使用和有效旗标

[0127] 未使用 [1:0] 和有效 [1:0] 旗标用于编码奇偶校验位的三个不同状态。下表列举这些状态。

[0128]

{未使用 [1:0]，有效 [1:0]}	奇偶状态
1111	未写入
0011	写入且有效
0000	失效

[0129] 然而，由于我们对于每个旗标只有两个冗余位，因而如果所述冗余位中的一者存在位错误，那么难以确定哪个冗余位是错误的。因此，我们将一直使用未使用 [1:0] 和有效 [1:0] 位两者来确定奇偶状态。下表展示通过计数那些位中 0 的总数目来进行此解码。

[0130]

0 数目	奇偶状态	{未使用 [1:0]，有效 [1:0]}
0	未写入	只有 1111
1	未写入	0111，1011，1101，1110
2	写入且有效	0011，0101，0110，1001，1010，1100
3	失效	100，0100，0010，0001
4	失效	只有 0000

[0131] 每个旗标使用两个冗余位而并非四个冗余位减少了存储器中所使用的位的数目，此代价在于冗余强度降低。

[0132] 特殊情况 - 页面 0

[0133] 当 TF_SECC_PAGE0_TB ＝'1'时，将略微不同地处理存储器页面 0。正常情况下，奇偶性 1 位可只在所述字的最后八字节中的任何位被编程时才被写入。在此特殊情况下，这个限制被取消了。因此，一旦奇偶性 0 位被使用时，下一次写入将一直导致奇偶性 1 位被使用。

[0134] 修整熔丝位

[0135] 有 3 种修整熔丝位会影响 ECC 逻辑：

[0136] TF_ECC_DISABLE_TB

[0137] 当 TF_ECC_DISABLE_TB ＝' 1' 时，禁用 ECC 逻辑且主机对每个页面的边带中的字节 512-527 具有完整读取和写入访问。

[0138] 当 TF_ECC_DISABLE_TB ＝' 0' 时，启用 ECC 逻辑且每个页面的边带中的字节 512-517 用于存储用于实施 LBECC 和智能 ECC 更新的各种 ECC 位。将丢弃针对边带区域的任何主机写入。来自边带区域的任何主机读取将返回不定数据。

[0139] TF_SMARTECC_DISABLE_TB

[0140] 当 TF_SMARTECC_DISABLE_TB ＝'1'时，禁用智能 ECC 更新逻辑。当 TF_SMARTECC_DISABLE_TB ＝'0'时，启用智能 ECC 更新逻辑。(TF_ECC_DISABLE_TB 必须也被设为'0'。)

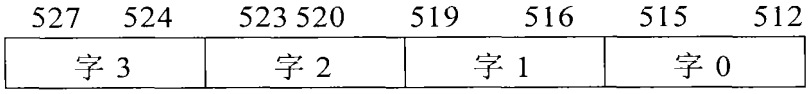
[0141] TF_SECC_PAGE0_TB

[0142] 当 TF_SECC_PAGE0_TB ＝'0'时，所有页面使用相同的智能 ECC 逻辑。当 TF_SECC_PAGE0_TB ＝' 1' 时，相对于智能 ECC 更新逻辑以不同于所有其它页面的方式来处理页面 0。

[0143] 存储器图

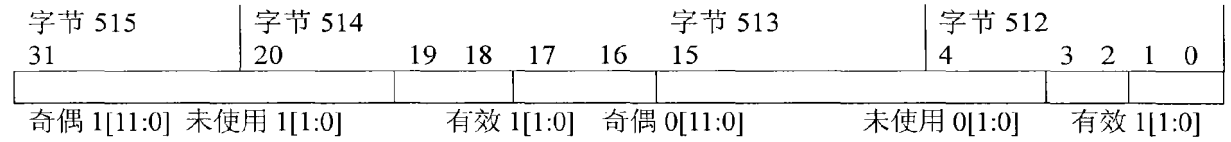
[0144] 所有 ECC 奇偶性、未使用和有效位都存在于每个页面的边带中的 512-527 字节中。下文展示所有四个 ECC 字的位是如何安排的。

[0145]



[0146] 下文展示针对一个 ECC 字是如何安排位的（实例针对字 0）。

[0147]



[0148] 测试

[0149] 可使用 DAM 模式来进行 ECC 逻辑的测试。当启用 DAM 模式来进行写入或读取交易时，将在页面寄存器与存储器阵列之间不传送任何数据。必须解除断言 TF_ECC_DISABLE_TB 位以让 ECC 逻辑被测试。下文更详细地描述此测试。

[0150] ECC 校正

[0151] 1. 启用 DAM 模式。启用 ECC。

[0152] 2. 发布 DATAIN 命令，所述命令具有 512 字节写入数据和 16 字节 ECC 数据。

[0153] 3. 发布 READ 命令。（由于在可读回数据之前进行 ECC 擦洗，芯片将会忙碌）。

[0154] 4. 验证 512 字节数据是如预期那样的。（如果将单个位错误写入到 PR 中，那么验

证所述错误已被校正。）

[0155] ECC 生成

[0156] 1. 启用 DAM 模式。启用 ECC。

[0157] 2. 发布 DATAIN 命令，所述命令具有 512 字节写入数据。

[0158] 3. 发布 PROG 命令。（这将 16 个生成的 ECC 字节写入到 PR 中。）

[0159] 4. 发布 READ 命令。（先前 512 字节写入数据将会与生成的 16 字节 ECC 数据一起被读回。）

[0160] 5. 验证 16ECC 字节是正确生成的。

[0161] 如上所述，一种使用 ECC 的方法使用 8B（八字节）ECC 字，其中 ECC 症状是 1B 宽。在一个实施例中，此造成 12.5％的额外开销且提供总晶粒大小的约 9-10％。此方法使用单位校正 / 双位检测算法，其并行处理 72 位数据。相反地，大块 ECC 使用 ECC 字，所述 ECC 字较大而足以减小症状存储的额外开销。如上文加注，目前优选使用单位检测唯一算法以进一步减小额外开销。此算法依次处理数据。

[0162] ECC 症状存储有各种选择。在一种选择（选择 1）中，ECC 症状位被存储在扩展页面区域中。一个目前优选的结构将 18B(16B 数据和 2B ECC 症状位）分配到扩展页面。由于此选择，修整熔丝或插针可用于 (1) 使用内部 ECC 生成（其中症状存储在扩展页面空间中）且用户不能访问扩展页面空间；且 (2) 绕过内部 ECC 生成以允许用户访问扩展页面空间并使用他们自己的 ECC 算法。在另一选择（选择 2）中，ECC 症状位被存储在专用空间中。根据块大小，此可使用某些 CR 空间。

[0163] 可在部分页面写入期间写入的最小量等于 ECC 字大小且当前在目前优选实施例中被设为 8B。存在若干种选择。在选择 1 中，存储器装置识别部份页面写入且针对所述页面禁用 ECC。对于此选择不存在逆向兼容性问题，然而，将损害部分写入之后的数据完善性。除了增加 N 次重写 ECC 能力之外，选择 2 与选择 1 相同。存储器装置将足够聪明来识别部分写入，使旧 ECC 症状失效，且将新症状写入到新位置。使用 128B 或更大的大块大小，此会能够为每个块 2-3 次重写提供足够的存储能力。在额外复杂性方面，这是对先前解决方案的改进。选择 3 提供智能 ECC 症状更新。在第一次写入到 ECC 字中期间，将写入所述症状。第二次写入到 ECC 字中将使所述症状失效，且数据将失去其 ECC 保护。在随后附加到 ECC 字之后，一旦其被完全写入，就会造成新的（且最终的）症状且数据从此将永远被保护。此显著减少了 DPM 暴露，因为仅有的未保护区域将是那些欠动态区域，所述区域并未全部由 ECC 字填充。除了增加冗余行修补能力之外，选择 4 与选择 2 相同。在选择 2 中 N 次重写之后，第 N+1 次重写触发行修补，且重写计数复位。

[0164] 在另一替代性方案中，提供在读取后数据擦洗技术。使用此技术，如果读取期间的前向应力是扰乱的主要原因，那么可检查 ECC 校正逻辑的状态且在位变坏时马上将所述坏位识别出来。接着，可使用现有 CR 方案来修补它。除非两个位同时变坏，否则这种方案是在读取扰乱之后进行的非常有效的方式。

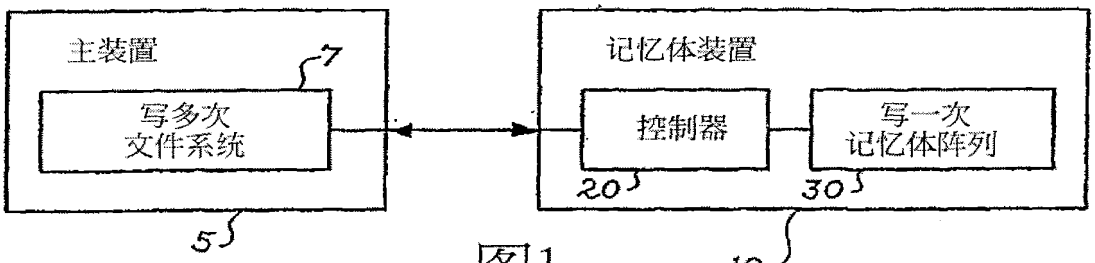

主装置
7
写多次
文件系统
5
记忆体装置
控制器
20
写一次
记忆体阵列
30
10

图1

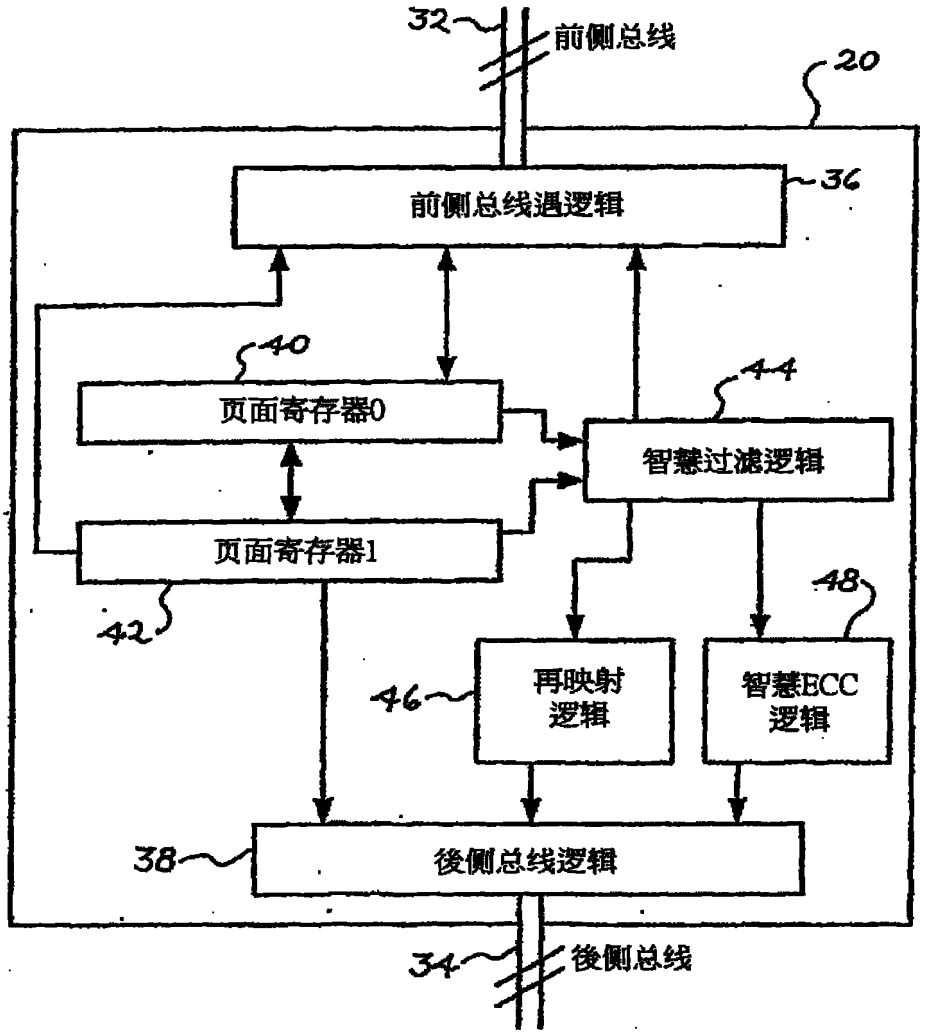

32
前侧总线
20
前侧总线遇逻辑
36
40
页面寄存器0
44
智慧过滤逻辑
页面寄存器1
42
48
46
再映射
逻辑
智慧ECC
逻辑
38
後侧总线逻辑
34
後侧总线

图 2

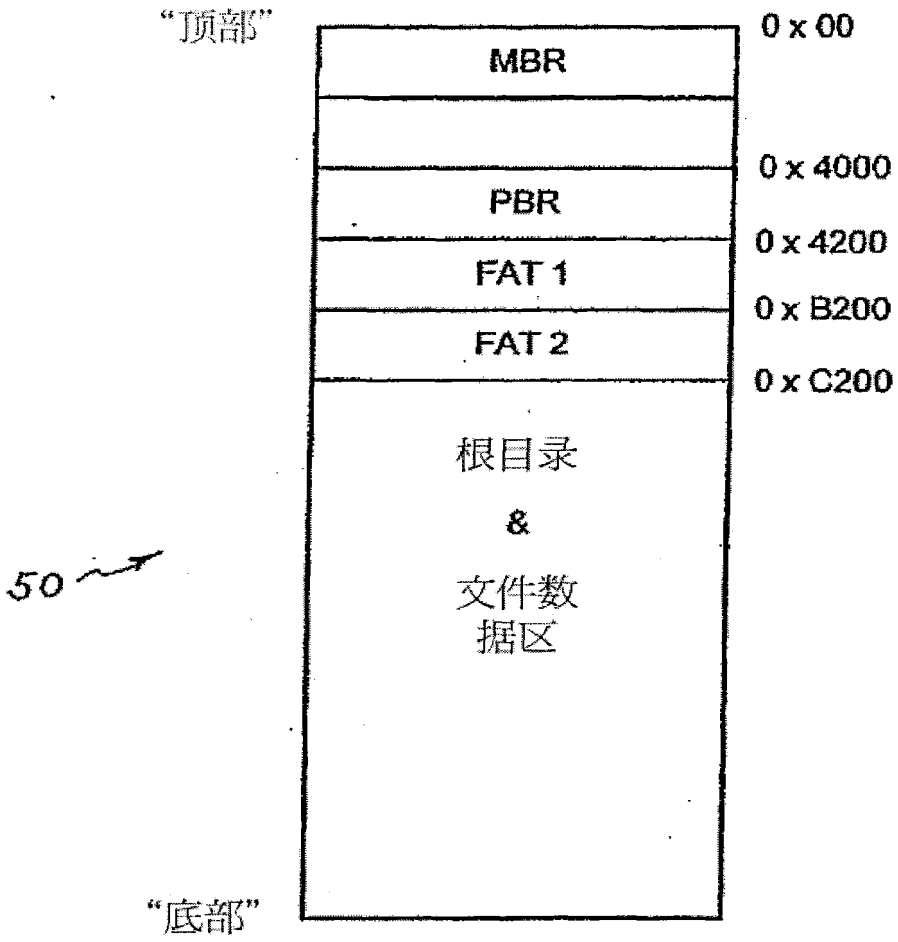

"顶部"
MBR
PBR
FAT 1
FAT 2
根目录
&
文件数
据区
"底部"
50
0 x 00
0 x 4000
0 x 4200
0 x B200
0 x C200

图 3

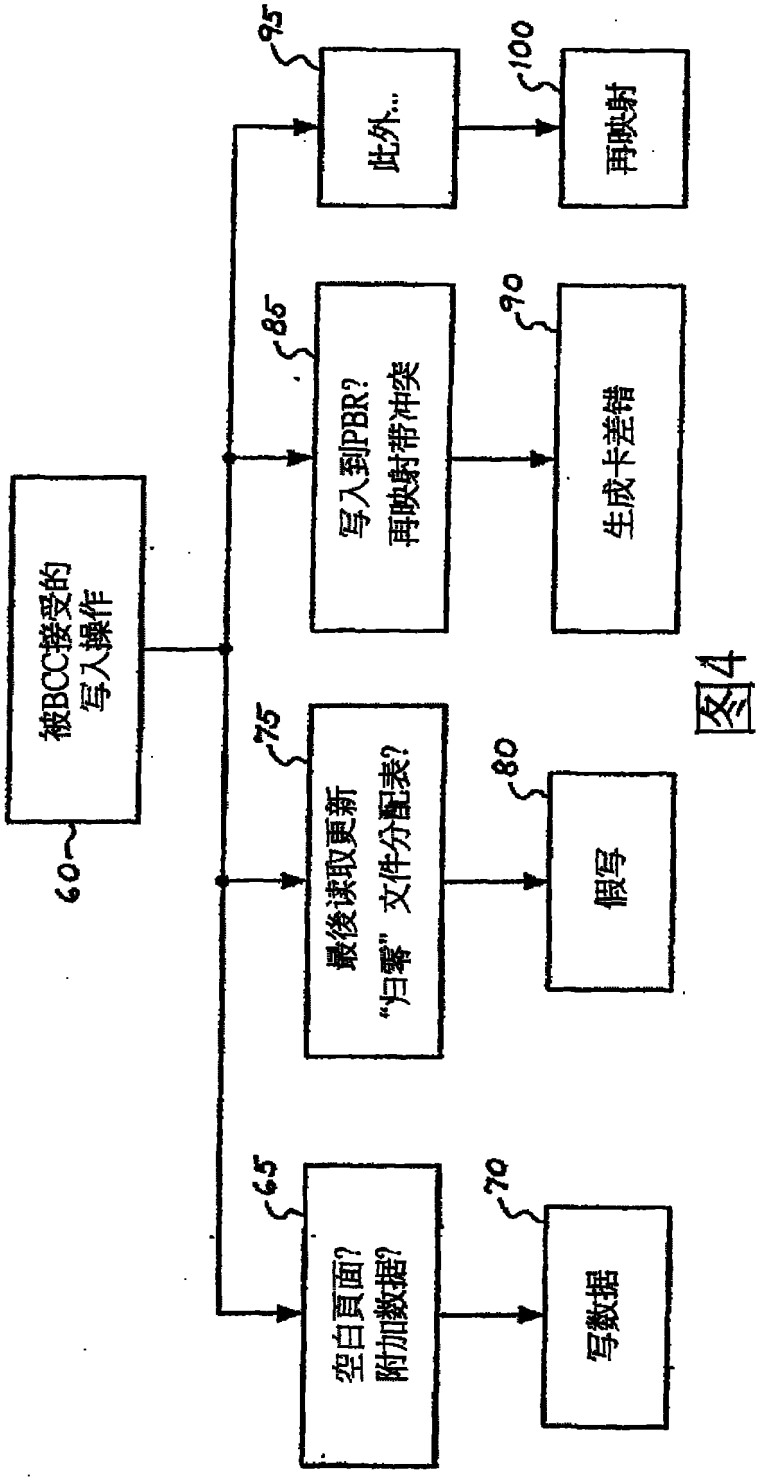

60
被BCC接受的写入操作
65
空白页面?附加数据?
70
写数据
75
最后读取更新"归零"文件分配表?
80
假写
85
写入到PBR?再映射带冲突
90
生成卡差错
95
此外...
100
再映射

图4

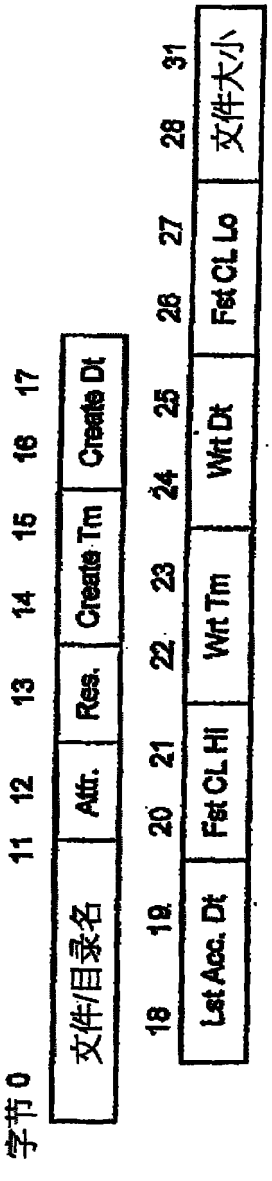

字节 0
11 12 13 14 15 16 17
文件/目录名
Attr.
Res.
Create Tm
Create Dt
18 19 20 21 22 23 24 25 26 27 28 31
Lst Acc. Dt
Fst CL Hi
Wrt Tm
Wrt Dt
Fst CL Lo
文件大小

图5

FF	FF	03	04
05	06	FF	08
FF	0A	0B	0C
0D	0E	0F	10
11	FF	00	00

图 6

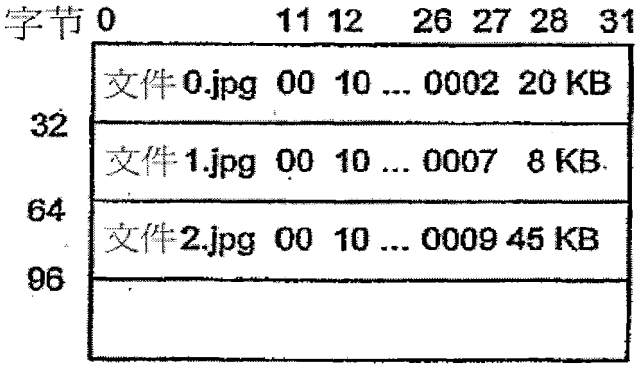


图 7

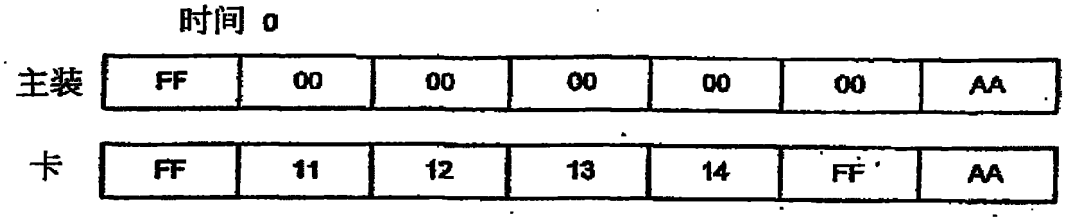


图 8

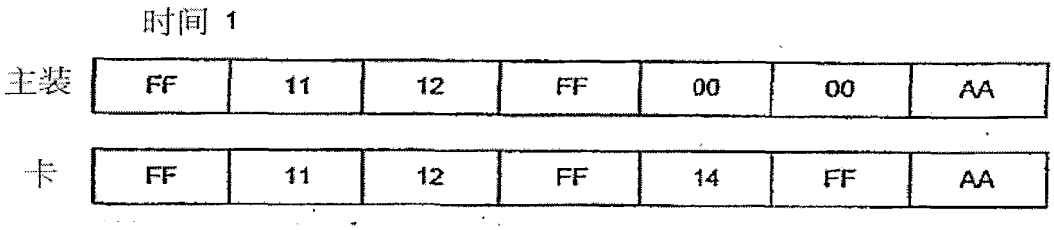


图 9

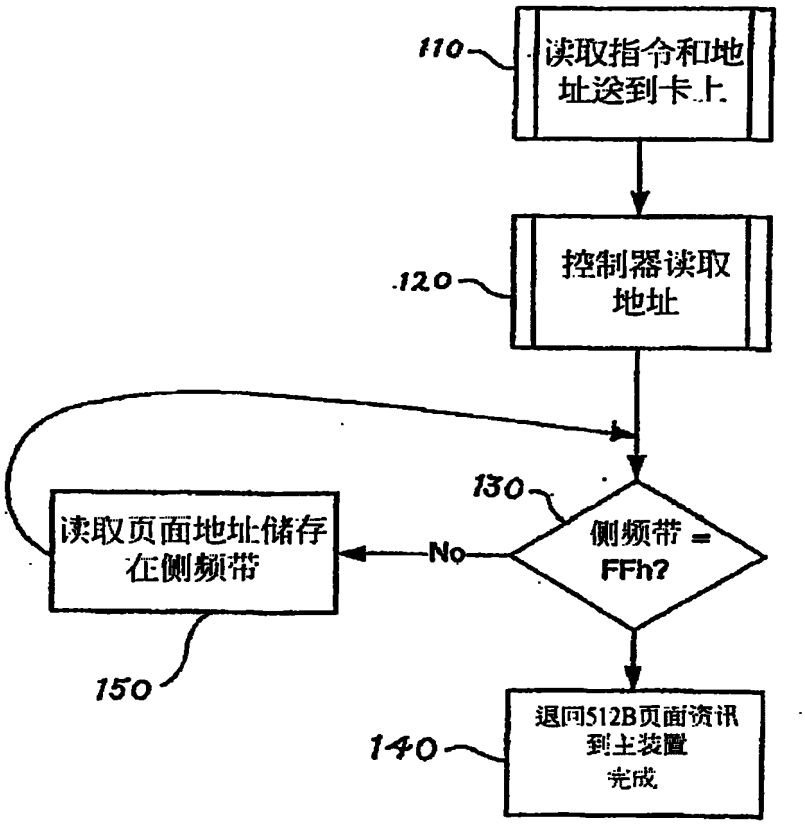

110
读取指令和地址送到卡上
120
控制器读取地址
130
侧频带 = FFh?
No
读取页面地址储存在侧频带
150
140
退回512B页面资讯到主装置
完成

图 10

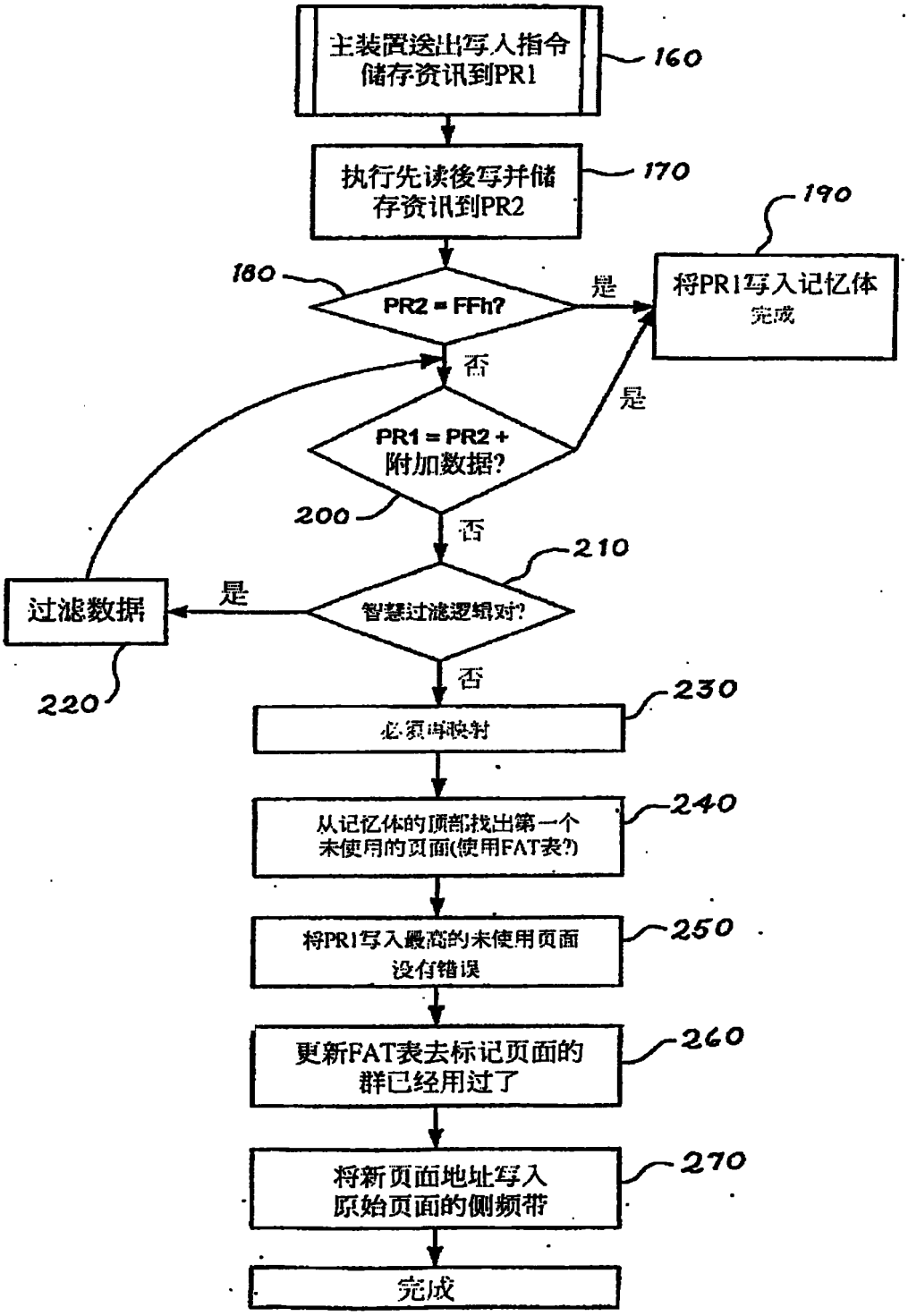

主装置送出写入指令
储存资讯到PR1
160
执行先读後写并储
存资讯到PR2
170
190
180
PR2 = FFh?
是
将PR1写入记忆体
完成
否
是
PR1 = PR2 +
附加数据?
200
否
210
过滤数据
是
智慧过滤逻辑对?
220
否
230
必须再映射
从记忆体的顶部找出第一个
未使用的页面(使用FAT表?)
240
将PR1写入最高的未使用页面
没有错误
250
更新FAT表去标记页面的
群已经用过了
260
将新页面地址写入
原始页面的侧频带
270
完成

图 11

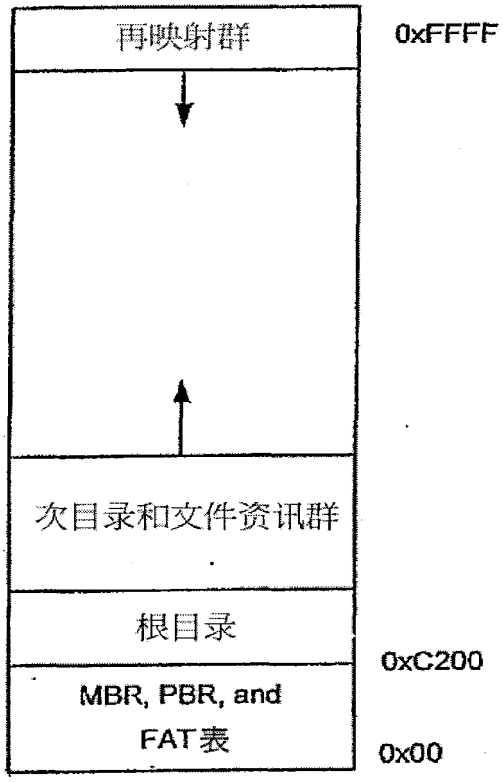

再映射群
0xFFFF
次目录和文件资讯群
根目录
0xC200
MBR, PBR, and
FAT表
0x00

图 12

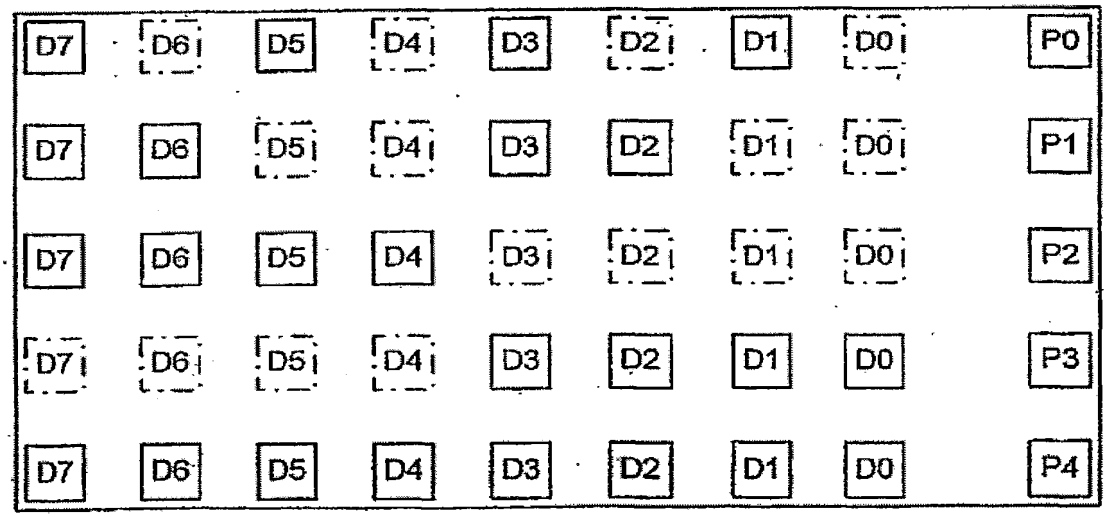

D7 D6 D5 D4 D3 D2 D1 D0 P0
D7 D6 D5 D4 D3 D2 D1 D0 P1
D7 D6 D5 D4 D3 D2 D1 D0 P2
D7 D6 D5 D4 D3 D2 D1 D0 P3
D7 D6 D5 D4 D3 D2 D1 D0 P4

图 13