



[12] 发明专利申请公布说明书

[21] 申请号 200680028166.8

[43] 公开日 2008年7月30日

[11] 公开号 CN 101233480A

[22] 申请日 2006.8.1

[21] 申请号 200680028166.8

[30] 优先权

[32] 2005.8.3 [33] US [31] 60/705,388

[32] 2006.5.8 [33] US [31] 60/746,742

[32] 2006.7.21 [33] US [31] 11/459,246

[32] 2006.7.21 [33] US [31] 11/459,255

[86] 国际申请 PCT/US2006/030098 2006.8.1

[87] 国际公布 WO2007/019175 英 2007.2.15

[85] 进入国家阶段日期 2008.1.31

[71] 申请人 桑迪士克股份有限公司

地址 美国加利福尼亚州

[72] 发明人 艾伦·W·辛克莱 巴里·赖特

[74] 专利代理机构 北京律盟知识产权代理有限责任
公司

代理人 刘国伟

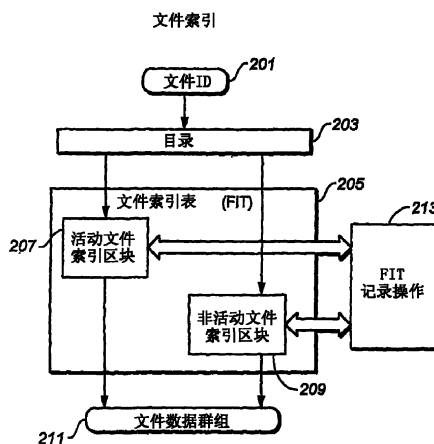
权利要求书5页 说明书37页 附图20页

[54] 发明名称

索引直接存储数据文件的可重新编程的非易失性存储器中的文件数据

[57] 摘要

用每个文件的唯一标识和所述文件内的数据偏移量将主机系统数据文件直接写入到大型擦除区块快闪存储器系统，而无需使用所述存储器的任何中间逻辑地址或虚拟地址空间。通过所述存储器系统的控制器而不是通过主机将关于所述文件存储在所述存储器中的哪个位置的目录信息保存在所述存储器系统内。每个数据文件在文件目录中唯一地识别，所述文件目录指向组成所述文件的数据群组的文件索引表(FIT)中的条目及所述数据群组在所述存储器中的物理存储位置。



1. 一种操作非易失性存储器系统的方法，所述存储器系统具有分组成存储器单元区块的存储单元，所述区块在重新编程有从所述存储器系统外部接收到的文件的数据之前被擦除，所述方法包括：

将具有唯一文件识别符的文件的数据存储在所述存储器区块中，
针对个别文件，保存所述个别文件的数据的物理区块位置的记录，
将所述个别文件的分类保存为（1）在不久的将来可能要修改或（2）在不久的将来不可能要修改，
保存通过文件的唯一文件识别符到达所述文件的所述记录的链接的目录，所述链接对于被分类为（1）的文件的记录是间接的，且对于被分类为（2）的文件的记录是直接的，以及
通过使用所述记录中的选定一者的唯一文件识别符，通过所述目录来存取所述选定一者。
2. 根据权利要求 1 所述的方法，其中当所述存储器系统连接到的主机系统将所述个别文件指定为打开时，将文件以分类（1）保存，且其中当所述存储器系统连接到的所述主机系统将所述个别文件指定为关闭时，将文件以分类（2）保存。
3. 根据权利要求 1 所述的方法，其中响应于从所述存储器系统附接到的主机在所述存储器区块中的至少一者内写入文件数据，所述存储器系统保存所述文件分类。
4. 根据权利要求 1 所述的方法，其中文件的所述间接链接包含不随着含有所述文件的所述记录的第二位置改变而改变的所述存储器中的第一位置的地址在所述目录中的条目，所述第一位置含有具有其任何改变的所述第二位置的地址，且其中文件的所述直接链接包含所述第二位置的地址在所述目录中的条目。
5. 根据权利要求 1 所述的方法，其中将分类为（1）的文件的记录保存在所述存储器区块内的里面没有其它文件的记录的个别页中，且其中将分类为（2）的文件的多个记录保存在所述存储器区块内的个别页中。
6. 根据权利要求 1 所述的方法，其中将分类为（2）的文件的若干记录一起分组在所述存储器区块的第一页中，且将到达那些数目的记录的若干链接一起分组在所述存储器区块的第二页中，从而形成所述目录。
7. 根据权利要求 6 所述的方法，其中将分类为（2）的文件的被存储在所述第一页中的所述若干记录限于链接存储在所述第二页中的文件。

8. 一种操作非易失性存储器系统的方法，所述存储器系统具有分组成存储器单元区块的存储单元，所述区块在重新编程有由唯一文件识别符识别的数据文件之前被擦除，所述方法包括：
 - 将个别文件的数据存储为一个或一个以上数据群组，
 - 保存组成所述数据文件的个别数据文件的所述数据群组的记录，以及
 - 保存通过所述唯一文件识别符到达所述记录的链接的目录，所述链接针对非活动文件是直接的，且针对活动文件是间接的。
9. 根据权利要求 8 所述的方法，其中活动文件是在不久的将来可能要修改的文件，且非活动文件是在不久的将来不可能要修改的文件。
10. 根据权利要求 8 所述的方法，其中活动文件包含由所述存储器系统连接到的主机指定为打开的文件，且非活动文件包含由所述主机指定为关闭的文件。
11. 根据权利要求 8 所述的方法，其中所述存储器系统通过监视所述个别文件在所述存储器区块内的至少一个存储模式来将所述文件标识为活动或非活动。
12. 根据权利要求 8 所述的方法，其中所述一个或一个以上数据群组包含存储器单元区块内的邻接逻辑文件偏移地址和邻接物理地址中的数据。
13. 根据权利要求 8 所述的方法，其中将活动文件的记录保存在所述存储器区块内的里面没有其它文件的记录的个别页中，且其中将非活动文件的多个记录保存在所述存储器区块内的个别页中。
14. 根据权利要求 8 所述的方法，其中将非活动文件的若干记录一起分组在所述存储器区块的第一页中，且将到达那些数目的记录的若干链接一起分组在所述存储器区块的第二页中，从而形成所述目录。
15. 一种操作非易失性存储器系统的方法，所述存储器系统具有分组成存储器单元区块的存储单元，所述区块在重新编程有由唯一文件识别符和所述文件内的偏移地址识别的数据之前被擦除，所述方法包括：
 - 将个别文件的数据存储为个别地指定存储器单元区块内的邻接逻辑文件偏移地址和邻接物理地址中的数据的一个或一个以上数据群组，
 - 将一组条目保存在所述个别文件的所述数据群组的表中，以及
 - 保存通过所述文件的唯一识别符将所述文件链接到所述文件的数据在所述表中的所述条目的文件目录，所述链接对于关闭文件的数据是直接的，且对于打开文件的数据是间接的。

16. 根据权利要求 15 所述的方法，其中将打开文件的表保存在所述存储器区块内的里面没有其它文件的表的个别页中，且其中将关闭文件的多个记录保存在所述存储器区块内的个别页中。
17. 根据权利要求 15 所述的方法，其中将关闭文件的若干记录一起分组在所述存储器区块的第一页中，且将到达那些数目的记录的若干链接一起分组在所述存储器区块的第二页中，从而形成所述目录。
18. 一种操作非易失性存储器系统的方法，所述存储器系统具有分组成存储器单元区块的存储单元，所述区块在重新编程有所述区块的页中的数据之前被擦除，所述方法包括：
 - 将个别地具有唯一文件识别符的文件的数据存储在第一个页中，
 - 将存储在所述第一个页中的所述文件的数据的物理页位置的记录保存在至少一第二页中，以及
 - 将指向至少所述第二页中的记录的所述位置的指针保存在至少一第三页中，其中所述第二页中的所述记录被限于指针存储在所述第三页中的所述文件的记录。
19. 根据权利要求 18 所述的方法，其中所述指向至少所述第二页中的记录的所述位置的指针与所述记录所属的所述文件的所述唯一识别符相关联。
20. 根据权利要求 18 所述的方法，其中将所述第一个页定位在一个或一个以上区块的第一群组中，将所述至少一第二页定位在一个或一个以上区块的第二群组中，且将所述至少一第三页定位在一个或一个以上区块的第三群组中。
21. 根据权利要求 18 所述的方法，其中将所述第一个页定位在一个或一个以上区块的第一群组中，且将所述至少一第二页和所述至少一第三页定位在一个或一个以上区块的第二群组的共用区块中。
22. 根据权利要求 18 所述的方法，其中将所述第一个页、所述至少一第二页和所述至少一第三页一起定位在一个或一个以上区块的群组的共用区块中。
23. 一种操作非易失性存储器系统的方法，所述存储器系统具有分组成存储器单元区块的存储单元，所述区块在将数据重新编程在所述区块的页中之前被擦除，所述方法包括：
 - 将存储有数据的物理页位置的记录保存在至少一第一页中，以及
 - 将指向至少所述第一页中的记录的所述位置的指针保存在至少一第二页中，其中所述第一页中的所述记录限于由存储在所述第二页中的所述指针指向的记录。

24. 一种操作非易失性存储器系统的方法，所述存储器系统具有分组成存储器单元区块的存储单元，所述区块在重新编程有所述区块的页中的文件的数据之前被擦除，所述方法包括：
- 将具有唯一文件识别符的文件的数据存储在所述存储器区块中，
 - 针对个别文件，保存所述个别文件的数据的物理区块位置的记录，
 - 保存通过文件的唯一文件识别符到达所述文件的所述记录的链接的目录，所述目录链接到达存储器的含有所述记录的物理地址的一个或一个以上位置，以及
 - 通过使用所述记录中的选定一者的唯一文件识别符，通过所述目录来存取所述选定一者，其中个别文件的记录保存在里面没有其它文件的记录的个别页中。
25. 一种非易失性存储器存储单元的存储器系统，所述存储单元分组成存储器单元区块，所述区块在重新编程有数据之前被擦除，其中：
- 从所述存储器系统外部接收到的具有唯一文件识别符的文件的数据通过其文件识别符存储在所述存储器区块中，
 - 数据的所述物理区块位置的记录针对所述个别文件而保存，
 - 所述个别文件的分类保存为（1）在不久的将来可能要修改或（2）在不久的将来不可能要修改，且
 - 到达所述文件的所述记录的链接的目录通过文件的唯一文件识别符保存，所述链接对于被分类为（1）的文件的记录是间接的，且对于被分类为（2）的文件的记录是直接的。
26. 根据权利要求 25 所述的存储器系统，进一步其中当所述存储器系统连接到的主机系统将所述个别文件指定为打开时，文件以分类（1）保存，且其中当所述存储器系统连接到的所述主机系统将所述个别文件指定为关闭时，文件以分类（2）保存。
27. 根据权利要求 25 所述的存储器系统，进一步其中文件的所述间接链接包含不随着含有所述文件的所述记录的第二位置改变而改变的所述存储器中的第一位置的地址在所述目录中的条目，所述第一位置含有具有任何改变的所述第二位置的地址，且其中文件的所述直接链接包括所述第二位置的地址在所述目录中的条目。
28. 根据权利要求 25 所述的存储器系统，进一步其中分类为（1）的文件的记录保存在所述存储器区块内的里面没有其它文件的记录的个别页中，且其中分类为（2）的文件的多个记录保存在所述存储器区块内的个别页中。
29. 根据权利要求 25 所述的存储器系统，进一步其中分类为（2）的文件的若干记录一

起分组在所述存储器区块的第一页中，且到达那些数目的记录的若干链接一起分组在形成所述目录的所述存储器区块的第二页中。

30. 根据权利要求 29 所述的存储器系统，进一步其中存储在所述第一页中的分类为(2)的文件的所述若干记录限于链接存储在所述第二页中的文件。
31. 一种非易失性存储器存储单元的存储器系统，所述存储单元分组成存储器单元区块，所述区块在将数据重新编程在所述区块的页中之前一起被擦除，其中：
 - 个别地具有唯一文件识别符的文件的数据存储在第一个页中，
 - 将所述唯一识别的文件的数据所在的所述物理页位置的记录保存在至少一第二页中，以及
 - 将指向所述至少一第二页中的记录的所述位置的指针保存在至少一第三页中，其中所述第二页中的所述记录限于指针存储在所述第三页中的所述文件的记录。
32. 根据权利要求 31 所述的存储器系统，进一步其中所述第一个页定位在一个或一个以上区块的第一群组中，所述至少一第二页定位在一个或一个以上区块的第二群组中，且所述至少一第三页定位在一个或一个以上区块的第三群组中。
33. 根据权利要求 31 所述的存储器系统，其中所述第一个页定位在一个或一个以上区块的第一群组中，且所述至少一第二页和所述至少一第三页定位在一个或一个以上区块的第二群组的共用区块中。
34. 根据权利要求 31 所述的存储器系统，其中所述第一个页、所述至少一第二页和所述至少一第三页一起定位在一个或一个以上区块的群组的共用区块中。

索引直接存储数据文件的可重新编程的非易失性存储器中的文件数据

技术领域

本申请案涉及例如半导体快闪存储器的可重新编程的非易失性存储器系统的操作，且更具体地说，涉及主机装置与存储器之间的接口的管理。

背景技术

对通过主机系统、存储器系统和其它电子系统的外部接口通信的数据进行寻址主要有两种技术。在其中的一种技术中，将系统产生或接收到的数据文件的地址映射到针对所述系统建立的连续逻辑地址空间的不同范围中。地址空间的广度通常足以覆盖系统能够处置的整个地址范围。在一个实例中，磁盘存储驱动器通过此逻辑地址空间与计算机或其它主机系统通信。此地址空间具有足以寻址磁盘驱动器的整个数据存储容量的广度。在所述两种技术中的第二种技术中，唯一地识别电子系统产生或接收到的数据文件，且通过所述文件内的偏移量来逻辑地寻址其数据。一种形式的这种寻址方法在计算机或其它主机系统与被称为“智能卡”的可移除存储卡之间使用。智能卡通常由消费者用来进行识别、银行业务、销售点购买、ATM接入和类似活动。

在早一代的商用快闪存储器系统中，将矩形存储器单元阵列分成大量单元群组，每个单元群组存储标准磁盘驱动器扇区的数据量，即512字节。额外量的数据（例如16个字节）通常也包含在每个群组中，以存储误差校正码（ECC）以及可能其它与用户数据且/或与里面存储有相关联的用户数据的存储器单元群组有关的额外开销数据。每个此群组中的存储器单元是可一起擦除的最小数目的存储器单元。即，擦除单位实际上是存储一个数据扇区和所包含的任何额外开销数据的若干存储器单元。这种类型的存储器系统的实例在第5,602,987号和第6,426,893号美国专利中描述。在使存储器单元重新编程有数据之前对其进行擦除是快闪存储器的特征。

快闪存储器系统最常见的是以可移除地与多种主机（例如个人计算机、相机或类似物）连接的存储卡或快闪驱动器的形式提供，但也可嵌入在此类主机系统内。当将数据写入到存储器时，主机通常将唯一逻辑地址指派给扇区、群集或存储器系统的连续虚拟地址空间内的其它数据单位。与磁盘操作系统（DOS）类似，主机将数据写入到存储器系统的逻辑地址空间内的地址并从所述地址读取数据。存储器系统内的控制器将从主机接收到的逻辑地址转换成实际存储数据的存储器阵列内的物理地址，且所述控制器接着

记住这些地址转换。存储器系统的数据存储容量至少与可在针对存储器系统界定的整个逻辑地址空间上寻址的数据的量一样大。

在后面几代快闪存储器系统中，擦除单位的大小增加到足以存储多个数据扇区的存储器单元的区块。即使与存储器系统连接的主机系统可以最小单位（例如扇区）编程和读取数据，但大量扇区存储在快闪存储器的单个擦除单位内。以下情况是常见的：当主机更新或替换逻辑数据扇区时，区块内的一些数据扇区变成废弃的。由于在区块中所存储的任何数据可被重写之前，必须擦除整个区块，所以新的或经更新的数据通常存储在已经被擦除且具有用于所述数据的剩余容量的另一区块中。这个过程留下具有废弃数据的原先区块，所述废弃数据占据存储器内的宝贵空间。但如果所述区块中还有任何有效数据留在里面，那么就不能擦除所述区块。

因此，为了更好地利用存储器的存储容量，常见的是通过将有效部分区块量的数据复制到经擦除的区块中来对其进行合并或收集，使得从中复制这些数据的区块接着可被擦除，且其整个存储容量可再用。还希望复制所述数据，以便以其逻辑地址的次序对区块内的数据扇区进行分组，因为这会提高读取所述数据和将所读取的数据传递到主机的速度。

如果此数据复制发生得太频繁，那么存储器系统的操作性能可能降级。这尤其会影响存储器系统的操作，其中存储器的存储容量几乎与可由主机通过系统的逻辑地址空间寻址的数据的量没有差别（典型情况）。在这种情况下，在可执行主机编程命令之前，可能需要数据合并或收集。于是编程时间因而增加。

在相继几代存储器系统中，区块的大小已经逐渐增加，以便增加可以存储在给定半导体区域中的数据位的数目。存储 256 个数据扇区和更多扇区的区块正变得越来越常见。另外，不同阵列或子阵列的两个、四个或更多区块通常逻辑上一起连接成元区块，以便提高数据编程和读取中的并行度。伴随此类大容量操作单位而来的是对其进行高效操作的难题。

用于此类存储器系统的常见主机接口是与通常与磁盘驱动器一起使用的接口类似的逻辑地址接口。存储器连接到的主机所产生的文件被指派有所述接口的逻辑地址空间内的唯一地址。存储器系统接着通常在逻辑地址空间与存储器的物理区块或元区块之间映射数据。存储器系统记住逻辑地址空间如何映射到物理存储器中，但主机不知道这个。主机记住其数据文件在逻辑地址空间内的地址，但存储器系统在不知道此映射的情况下操作。

发明内容

已经开发了很多技术，其在各种程度上克服了高效地操作此类大型擦除区块快闪存储器系统中所遇到的问题。另一方面，本发明是基于基本变化的，即通过改变存储器与主机系统之间的数据传递接口。不是通过使用虚拟地址空间内的逻辑地址来在它们之间传送数据，而是数据文件通过主机所指派的文件名来识别，且其由所述文件内的偏移地址存取。接着，存储器系统知道每个扇区或其它数据单位所属的主机文件。本文论述的文件单位是（例如）通过具有循序偏移地址而排序的一组数据，且所述组数据由在主机计算系统中操作的应用程序创建并唯一地识别。

这并不被大多数当前商用存储器系统所使用，因为主机现在在不识别文件的情况下，通过一组共用逻辑地址来在所有文件内向存储器系统识别数据。通过由文件对象而不是使用逻辑地址来识别主机数据，存储器系统控制器可以用减少对此频繁数据合并和垃圾收集的需要的的方式存储数据。数据复制操作的频率和所复制数据的量因此显著减少，从而提高了存储器系统的数据编程和读取性能。另外，存储器系统控制器保存主机文件存储在其中的存储器区块的目录和索引表信息。于是，主机没有必要保存当前对管理逻辑地址接口来说必要的文件分配表（FAT）。

相反，文件的数据由数据群组识别，且文件的数据群组在存储器内的位置保存在文件索引表（FIT）中。如上文交叉参考的专利申请案（尤其是第 11/060,248 号专利申请案）中更全面地描述，数据群组含有具有邻接逻辑偏移地址和邻接物理存储器地址两者的数据。如也在上文交叉参考的专利申请案中描述，每个文件的有效数据群组的索引根据其偏移地址保存在 FIT 中，借助使用其唯一文件识别符，通过文件目录来针对给定文件存取所述偏移地址。对文件的 FIT 条目的存取接着给出所述文件的数据群组的物理位置。

存储器系统可与其连接的主机通常以一次打开的文件具有预设最大数目（例如五个此类文件）的方式操作。在需要在已经打开了最大数目的文件时打开新的文件的情况下，当前打开的文件首先由主机关闭。当通过将额外数据写入到文件、使其数据中的一些更新或删除等等来修改所述文件时，打开文件的数据群组经受频繁改变。更新文件的数据群组的 FIT 列表通常要求读取含有所述文件的条目的页、修改那些条目且接着将经修改的页重写到存储器的不同位置中，通常是同一存储器区块内的经擦除页（如果存在一个的话）。因此，文件目录需要能够在含有文件的 FIT 条目的当前页改变时指向所述页。

优选的是，能够在不需要重写文件目录中的任何内容的情况下，更新活动文件的数据群组的列表。间接寻址打开文件的数据群组的技术允许文件目录在存取文件的 FIT 条

目时，保持指向存储器中的同一位置。文件目录优选寻址包含在 FIT 条目的已经写入的最后一个页中的逻辑指针，所述逻辑指针含有具有所述文件的经更新的 FIT 条目的页的地址。尽管当文件的 FIT 条目更新时逻辑指针改变，但所述文件的文件目录中的地址无需改变。

然而，当大量文件存储在存储器系统中时，此类间接寻址的效率不高（例如在许多小型文件被存储时发生），因为所述目录和 FIT 页一次只能含有这么多文件的条目，当然这视其数据存储容量而定。因此，对于由主机关闭且因此不是当前活动的文件来说，优选通过文件目录直接寻址索引条目。不使用作为间接寻址的特征的逻辑地址指针。

对于可能在不久的将来会改变的文件（例如打开文件）的 FIT 条目的间接寻址与关闭文件的直接寻址的组合因此是优选的操作模式。即，对于数据活动地写入或更新到的打开文件来说，文件目录以可在不需要修改文件目录的情况下修改活动文件的一组索引条目的方式，间接地界定由文件的唯一识别符识别的所述组索引条目在 FIT 中的位置。在特定实例中，可通过寻址存储器中将其参考到文件的索引条目的实际物理位置的给定位置的文件目录来实现上述情况。此参考在索引目录更新和移动时改变，但不必更新文件目录。同时，对于关闭文件来说，文件目录直接界定由唯一文件识别符识别的一组索引条目在 FIT 中的位置。与打开文件相比，关闭文件的索引条目的重新定位的可能性小得多。

另外，为了在直接寻址中使用，FIT 页中的索引条目优选限于其指向 FIT 的指针包含在文件目录的同一页中的文件的数据群组。因此，在所述目录页上所列出的文件的一个或一个以上 FIT 条目被更新时，只有文件目录的一个页需要更新。否则，当文件的一个或一个以上 FIT 条目被更新时，通常将有必要更新多个文件目录页。

本发明的其它方面、优势、特征和细节包含在以下对本发明的示范性实例的描述内容中，所述描述内容应结合附图来理解。

本文所参考的所有专利、专利申请案、文章、其它出版物、文献等等之类出于所有目的全文以引用的方式并入本文中。如果所并入的任何出版物、文献等等之类与本申请案之间在术语的定义或用法方面存在任何不一致性或冲突，则应以本申请案的定义或用法为准。

附图说明

图 1 示意性地说明如当前所实施的主机和所连接的非易失性存储器系统；

图 2 是用作图 1 的非易失性存储器的示范性快闪存储器系统的方框图；

- 图 3 是可在图 2 的系统中使用的存储器单元阵列的代表性电路图；
- 图 4 说明图 2 的系统的示范性物理存储器组织；
- 图 5 展示图 4 的物理存储器的一部分的展开图；
- 图 6 展示图 4 和图 5 的物理存储器的一部分的进一步展开图；
- 图 7 说明主机与可重新编程的存储器系统之间的常见现有技术逻辑地址接口；
- 图 8 以与图 7 不同的方式说明主机与可重新编程的存储器系统之间的常见现有技术逻辑地址接口；
- 图 9 说明根据本发明的主机与可重新编程的存储器系统之间的直接文件存储接口；
- 图 10 以与图 9 不同的方式说明根据本发明的主机与可重新编程的存储器系统之间的直接文件存储接口；
- 图 11 是与直接数据文件存储接口一起操作的存储器系统的功能方框图；
- 图 12 说明直接数据文件存储器的操作循环；
- 图 13A 到图 13D 展示将文件的数据直接写入到存储器中的四个不同实例；
- 图 14A 到图 14E 说明将单个数据文件直接写入到存储器中的序列；
- 图 15 展示回收图 14E 的区块的结果；
- 图 16 说明回收只含有一个数据群组和废弃数据的区块；
- 图 17 展示图 14A、图 14C、图 14E、图 15 和图 16 的数据群组的 FIT 的条目；
- 图 18 说明第一特定文件索引实施例的使用活动和非活动文件的不同 FIT 结构的文件索引的原理；
- 图 19 扩展图 18 的说明内容；
- 图 20 给出图 18 和图 19 的文件目录的示范性逻辑结构；
- 图 21 说明含有图 18 和图 19 的文件目录的存储器区块的物理结构；
- 图 22 展示图 18 和图 19 的 FIT 的逻辑结构；
- 图 23 说明含有活动文件的 FIT 条目 (TFIT) 的区块的物理结构；
- 图 24 说明含有非活动文件的 FIT 条目的区块的物理结构；
- 图 25 展示图 19 的文件索引实例中对 FIT 记录执行的操作；
- 图 26 说明根据第二特定文件索引实施例的文件索引的分级结构；以及
- 图 27 展示专用于图 26 的实施例中的文件目录、FIT 和文件属性表的存储器区块中的示范性数据结构。

具体实施方式

快闪存储器系统一般说明

相对于图 1 到图 8 描述常见快闪存储器系统和与主机装置的典型操作。在此类系统中，可实施本发明的各个方面。图 1 的主机系统 1 将数据存储在与快闪存储器 2 中，并从快闪存储器 2 检索数据。尽管快闪存储器可嵌入在主机内，但将存储器 2 说明为呈现更为普遍的通过机械和电连接器的配合部分 3 和 4 可移除地连接到主机的卡的形式。目前有许多不同的商业上可购得的快闪存储卡，实例是那些以 CompactFlash (CF)、MultiMediaCard (MMC)、Secure Digital (SD)、miniSD、microSD、Memory Stick、SmartMedia 和 TransFlash 的商标出售的存储卡。尽管这些卡具有根据其标准化规格的唯一机械和/或电接口，但每个卡中所包含的快闪存储器是非常相似的。这些卡都可从本申请案的受让人 SanDisk 公司购得。SanDisk 还在其 Cruzer 商标下提供一系列快闪驱动器，其为呈小型封装形式的手持式存储器系统，所述系统具有用于通过插入到主机的 USB 插口中而与主机连接的通用串行总线 (USB) 插头。这些存储卡和快闪驱动器中的每一者均包含控制器，所述控制器与主机介接，并控制它们内的快闪存储器的操作。

使用此类存储卡和快闪驱动器的主机系统有许多且是各式各样的。它们包含个人计算机 (PC)、膝上型和其它便携式计算机、蜂窝式电话、个人数字助理 (PDA)、数字照相机、数字摄影机和便携式音频播放器。主机通常包含内建式插口以供一种或一种以上类型的存储卡或快闪驱动器使用，但有些需要存储卡插入到其中的适配器。存储器系统通常含有其自己的存储器控制器和驱动器，但也存在一些只有存储器的系统，其改为由所述存储器连接到的主机所执行的软件来控制。在含有控制器的一些存储器系统 (尤其是那些嵌入在主机内的存储器系统) 中，存储器、控制器和驱动器通常形成于单个集成电路芯片上。

就存储器 2 涉及到的范围而言，图 1 的主机系统 1 可被视为具有两个主要部分，由电路和软件的组合构成。所述两个主要部分是应用程序部分 5 和与存储器 2 介接的驱动器部分 6。举例来说，在个人计算机中，应用程序部分 5 可包含运行文字处理、图形、控制或其它普及应用软件的处理器。在摄像机、蜂窝式电话或其它主要专用于执行单组功能的主机系统中，应用程序部分 5 包含操作摄像机以拍摄并存储图片、操作蜂窝式电话以发出和接收呼叫等等的软件。

图 1 的存储器系统 2 包含快闪存储器 7 和电路 8，所述两者都与卡所连接到的主机介接，以用于来回传递数据并控制存储器 7。控制器 8 通常在数据编程和读取期间在主机 1 所使用的数据的逻辑地址与存储器 7 的物理地址之间转换。

参看图 2, 描述可用作图 1 的非易失性存储器 2 的典型快闪存储器系统的电路。通常在通过系统总线 13 与一个或一个以上集成电路存储器芯片并联连接的单个集成电路芯片 11 上实施系统控制器, 图 2 中展示单个此类存储器芯片 15。所说明的特定总线 13 包含单独一组导线 17 用来运载数据、一组导线 19 用于存储器地址以及一组导线 21 用于控制与状态信号。或者, 单组导线可在这三个功能之间时间共享。另外, 可使用系统总线的其它配置, 例如 2004 年 8 月 9 日申请的第 10/915,039 号美国专利申请案(现第 2006/0031593 A1 号公开案)中所描述的环形总线。

典型的控制器芯片 11 具有其自己的内部总线 23, 所述内部总线 23 通过接口电路 25 与系统总线 13 介接。通常连接到总线的主要功能是处理器 27 (例如微处理器或微控制器); 只读存储器(ROM)29, 其含有用以初始化(“引导”)系统的代码; 只读存储器(RAM)31, 其主要用于缓冲正在存储器与主机之间传递的数据; 以及电路 33, 其计算并检查经过存储器与主机之间的控制器的数据的误差校正码(ECC)。控制器总线 23 通过电路 35 与主机系统介接, 在图 2 的系统包含在存储卡内的情况下, 这是通过卡的作为连接器 4 的一部分的外部触点 37 来完成的。时钟 39 与控制器 11 的其它组件中的每一者连接并由其利用。

存储器芯片 15 以及与系统总线 13 连接的任何其它芯片通常含有组织成多个子阵列或平面的存储器单元阵列, 为了简单起见, 说明两个此类平面 41 和 43, 但可改为使用更多(例如四个或八个)此类平面。或者, 芯片 15 的存储器单元阵列可不分成平面。然而, 当这样分时, 每个平面具有其自己的列控制电路 45 和 47, 其可很大程度上独立于彼此而操作。电路 45 和 47 从系统总线 13 的地址部分 19 接收其相应存储器单元阵列的地址, 并对所述地址进行解码以寻址相应位线 49 和 51 中的特定一者或一者以上。通过行控制电路 55 响应于在地址总线 19 上接收到的地址而寻址字线 53。源极电压控制电路 57 和 59 也与相应的平面连接, p 阱电压控制电路 61 和 63 也是如此。如果存储器芯片 15 具有单个存储器单元阵列, 且如果系统中存在两个或两个以上此类芯片, 那么每个芯片的阵列可以与上文所述的多平面芯片内的一个平面或子阵列类似的方式操作。

数据通过与系统总线 13 的数据部分 17 连接的相应数据输入/输出电路 65 和 67 转移到平面 41 和 43 中且从平面 41 和 43 中转移出来。电路 65 和 67 用于通过线 69 和 71 将数据编程到存储器单元中和从其相应平面的存储器单元中读取数据两者, 所述线 69 和 71 通过相应的列控制电路 45 和 47 连接到平面。

尽管控制器 11 控制存储器芯片 15 的操作以编程数据、读取数据、擦除和办理各种

内务处理事项，但每个存储器芯片还含有一些控制电路，所述控制电路执行来自控制器 11 的命令以执行此类功能。接口电路 73 连接到系统总线 13 的控制与状态部分 21。将来自控制器的命令提供到状态机 75，状态机 75 接着提供其它电路的特定控制，以便执行这些命令。控制线 77 到 81 使状态机 75 与这些其它电路连接，如图 2 中所示。来自状态机 75 的状态信息通过线 83 传送到接口 73，以便通过总线部分 21 传输到控制器 11。

存储器单元阵列 41 和 43 的 NAND 结构当前是优选的，但是也可改为使用其它结构（例如 NOR）。NAND 快闪存储器及其作为存储器系统的一部分的操作的实例可通过参考第 5,570,315 号、第 5,774,397 号、第 6,046,935 号、第 6,373,746 号、第 6,456,528 号、第 6,522,580 号、第 6,771,536 号和第 6,781,877 号美国专利以及第 2003/0147278 号美国专利申请公开案获得。

图 3 的电路图说明示范性 NAND 阵列，其为图 2 的存储器系统的存储器单元阵列 41 的一部分。提供大量全局位线，为了阐释的简单性，图 2 中仅展示四个此类线 91 到 94。许多串联连接的存储器单元串 97 到 104 连接在这些位线中的一者与参考电位之间。使用存储器单元串 99 作为代表，多个电荷存储存储器单元 107 到 110 在所述串的一端处与选择晶体管 111 和 112 串联连接。当串的选择晶体管变得具有传导性时，所述串连接在其位线与参考电位之间。接着，一次对所述串内的一个存储器单元进行编程或读取。

图 3 的字线 115 到 118 个别地延伸越过许多存储器单元串中的每一者中的一个存储器单元的电荷存储元件，且栅极 119 和 120 控制所述串的每个端处的选择晶体管的状态。使共享共用字线和控制栅极线 115 到 120 的存储器单元串形成一起擦除的存储器单元区块 123。此单元块含有一次可物理擦除的最小数目的单元。一次对沿字线 115 到 118 中的一者的一行存储器单元进行编程。通常，以指定次序对 NAND 阵列的行进行编程，在此情况下，开始于沿最接近于连接到接地或另一共用电位的串的端部的字线 118 的行。接下来对沿字线 117 的那行存储器单元进行编程，以此类推，贯穿区块 123。最后对沿字线 115 的行进行编程。

第二区块 125 是类似的，其存储器单元串连接到与第一区块 123 中的串相同的全局位线，但具有不同组的字线和控制栅极线。通过行控制电路 55 将字线和控制栅极线驱动到其合适的操作电压。如果系统中存在一个以上平面或子阵列，例如图 2 的平面 1 和 2，那么一个存储器结构使用在它们之间延伸的共用字线。或者，可能存在两个以上共享共用字线的平面或子阵列。在其它存储器结构中，分别驱动个别平面或子阵列的字线。

如上文参考的 NAND 专利和公开申请案中的几者中所述，存储器系统可经操作以在

每个电荷存储元件或区域中存储两个以上可检测电荷电平，从而在每一者中存储一个以上数据位。存储器单元的电荷存储元件最通常是导电性浮动栅极，但可替代地是非导电性介电电荷俘获材料，如第 2003/0109093 号美国专利申请公开案中所述。

图 4 在概念上说明用作下文进一步描述内容中的实例的快闪存储器单元阵列 7(图 1)的组织。存储器单元的四个平面或子阵列 131 到 134 可位于单个集成存储器单元芯片上、位于两个芯片（每个芯片上有所述平面中的两者）或位于四个单独芯片上。特定布置对下文的论述内容并不重要。当然，系统中可能存在其它数目的平面，例如 1 个、2 个、8 个、16 个或更多。通过矩形将所述平面个别地分成图 4 中所示的存储器单元区块，例如位于相应平面 131 到 134 内的区块 137、138、139 和 140。每个平面内可能存在数十个或数百个区块。如上文所提及，存储器单元区块是擦除单位，即可一起物理擦除的最小数目的存储器单元。然而，为了获得增加的并行性，以较大的元区块单位来操作所述区块。来自每个平面的一个区块逻辑上连接在一起以形成元区块。展示四个区块 137 到 140 形成一个元区块 141。元区块内的所有单元通常一起擦除。用于形成元区块的区块无需限于其相应平面内的相同相对位置，如由区块 145 到 148 组成的第二元区块 143 中所示。尽管通常优选使元区块延伸越过所有平面，但为了获得较高系统性能，可用动态地形成位于不同平面内的一个、两个或三个区块中的任何一者或所有的元区块的能力来操作存储器系统。这允许元区块的大小与一个编程操作中可用于存储的数据的量更紧密地匹配。

出于操作目的，个别区块又分成存储器单元页，如图 5 中所说明。举例来说，区块 131 到 134 中的每一者的存储器单元每一者被分成八个页 P0 到 P7。或者，每个区块内可存在 16 个、32 个或更多个存储器单元页。页是区块内的含有一次所编程的最小量的数据的数据编程和读取单位。在图 3 的 NAND 结构中，页由区块内的沿字线的存储器单元形成。然而，为了增加存储器系统操作并行性，两个或两个以上区块内的此类页可逻辑上连接成元页。图 5 中说明元页 151，其由来自四个区块 131 到 134 中的每一者的一个物理页形成。举例来说，元页 151 包含四个区块中的每一者中的页 P2，但元页的页不一定需要在区块的每一者内具有同一相对位置。

尽管优选越过所有四个平面并行编程和读取最大量的数据，但为了获得较高系统性能，存储器系统也可经操作以形成不同平面内的单独区块中的一个、两个或三个页中的任何一者或所有的元页。这允许编程和读取操作适应性地与可方便地并行处置的数据的量匹配，且减少元页的一部分保持未编程有数据的机会。

如图 5 中所说明，由多个平面的物理页形成的元页沿那多个平面的字线行含有存储

器单元。不是同时编程一个字线行中的所有单元，而是更通常在两个或两个以上交错的群组中交替地对它们进行编程，每个群组存储一个数据页（在单个区块中）或一个数据元页（越过多个区块）。通过一次对交替的存储器单元进行编程，无需为每个位线提供包括数据寄存器和读出放大器的外围电路单元，而是在邻近位线之间对其进行时间共享。这节约了外围电路所需的衬底空间的量，且允许存储器单元沿行以增加的密度堆积。否则，优选沿行同时编程每一单元，以便使可从给定存储器系统得到的并行性最大。

参看图 3，通过沿 NAND 串的至少一个端提供两行选择晶体管（未图示）而不是所示的单个行，来最方便地实现沿行同时将数据编程到每隔一个存储器单元中。一个行的选择晶体管接着响应于一个控制信号将区块内的每隔一个串连接到其相应的位线，且另一行的选择晶体管响应于另一控制信号将介入的每隔一个串连接到其相应的位线。因此，将两个数据页写入到每个存储器单元行中。

按照惯例，每个逻辑页中的数据的数据量通常是一个或一个以上数据扇区的整数数目，每个扇区含有 512 个字节的数据。图 6 展示页或元页的两个数据扇区 153 和 155 的逻辑数据页。每个扇区通常含有：部分 157，其具有正被存储的 512 个字节的用户或系统数据；和另一数目的字节 159，其用于与部分 157 中的数据有关或与其存储在里面的物理页或区块有关的额外开销数据。额外开销数据的字节数目通常是 16 个字节，使得扇区 153 和 155 中的每一者总共具有 528 个字节。额外开销部分 159 可含有在编程期间从数据部分 157 计算出的 ECC、其逻辑地址、区块已经被擦除和重新编程的次数的经历计数、一个或一个以上控制旗标、操作电压电平和/或类似物，加上从此类额外开销数据 159 计算出的 ECC。或者，额外开销数据 159 或其一部分可存储在其它区块中的不同页中。

随着存储器的并行性增加，元区块的数据存储容量增加，且因此数据页和元页的大小也增加。数据页于是可含有两个以上数据扇区。在一个数据页中具有两个扇区且每个元页具有两个数据页的情况下，一个元页中存在四个扇区。因此，每个元页存储 2048 个字节的数据。这是高度的并行性，且可能随着行中的存储器单元的数目增加而更进一步增加。为此，快闪存储器的宽度延伸，以便增加页和元页中的数据的数据量。

上文指出的物理上较小的可重新编程的非易失性存储卡和快闪驱动器商业上可以 512 兆字节（MB）、1 千兆字节（GB）、2GB 和 4GB 的数据存储容量购得，且可变得更高。图 7 说明主机与此大容量存储器系统之间的最常见的接口。主机处理由主机执行的应用软件或固件程序所产生或使用的数据文件。文字处理数据文件是一个实例，且计算机辅助设计（CAD）软件的绘图文件是另一个实例，主要用于例如 PC、膝上型计算机和

类似物的通用计算机主机中。pdf 格式的文档也是这样的文件。静止数字视频摄像机针对每个图片产生一个存储在存储卡上的数据文件。蜂窝式电话利用来自内部存储卡上的文件（例如电话号码簿）的数据。PDA 存储并使用若干不同文件，例如地址文件、日历文件和类似文件。在任一此类应用中，存储卡还可含有操作主机的软件。

图 7 中说明主机与存储器系统之间的常见逻辑接口。连续的逻辑地址空间 161 足够大以便为可存储在存储器系统中的所有数据提供地址。通常将主机地址空间分成数据群集的增量。每个群集可在给定主机系统中设计成含有若干数据扇区，大约在 4 到 64 个扇区是典型的。标准扇区含有 512 个字节的数据。

图 7 的实例中展示已经创建了三个文件 1、2 和 3。在主机系统上运行的应用程序创建每个文件作为一组有序数据，并通过唯一名称或其它参考对其进行识别。尚未分配给其它文件的足够可用的逻辑地址空间由主机指派给文件 1。文件 1 被展示为已经被分配有连续范围的可用逻辑地址。地址的范围通常还出于特定目的而分配，例如特定范围用于主机操作软件，所述地址接着避免用于存储数据，即使在主机将逻辑地址指派给数据时这些地址尚未被利用也是如此。

当主机稍后创建文件 2 时，主机类似地在逻辑地址空间 161 内指派两个不同范围的连续地址，如图 7 中所示。文件无需被指派有连续的逻辑地址，而是可为已经分配给其它文件的地址范围之间的地址片段。此实例接着展示主机所创建的又一文件 3 被分配有主机地址空间的先前未分配给文件 1 和 2 和其它数据的其它部分。

主机通过保存文件分配表（FAT）来记住存储器逻辑地址空间，其中由主机通过转换 160 指派给各个主机文件的逻辑地址被保存。随着新文件被存储、其它文件被删除、文件被修改等等，FAT 表由主机频繁地更新。FAT 表通常存储在主机存储器中，FAT 表的备份也存储在非易失性存储器中并被不时地更新。正如任何其它数据文件一样，通常通过逻辑地址空间在非易失性存储器中存取所述备份。当主机文件被删除时，主机接着通过更新 FAT 表来解除分配先前分配给被删除文件的逻辑地址，以展示它们现在可用于与其它数据文件一起使用。

主机不关心存储器系统控制器选择用来存储文件的物理位置。典型的主机仅知道其逻辑地址空间和其已经分配给其各个文件的逻辑地址。另一方面，存储器系统通过典型的主机/卡接口，仅知道逻辑地址空间的数据已经被写入到的部分，但不知道分配给特定主机文件的逻辑地址，乃至不知道主机文件的数目。存储器系统控制器将主机所提供的用于存储或检索数据的逻辑地址转换成存储有主机数据的快闪存储器单元阵列内的唯一

物理地址。区块 163 表示这些逻辑到物理地址转换的工作表，其由存储器系统控制器保存。

存储器系统控制器经编程以便以使系统的性能维持在高等级的方式将数据文件存储在存储器阵列 165 的区块和元区块内。在此说明中使用四个平面或子阵列。优选越过由来自所述平面中的每一者的区块形成的整个元区块，以系统允许的最大并行度对数据进行编程和读取。通常将至少一个元区块 167 分配为用于存储操作固件和存储器控制器所使用的数据的保留区块。可分配另一元区块 169 或多个元区块用于存储主机操作软件、主机 FAT 表等等。物理存储空间的大部分留下用于存储数据文件。然而，存储器控制器不知道接收到的数据如何已经由主机在其各个文件对象之间分配。存储器控制器通常从与主机的交互中知道的只是由主机写入到特定逻辑地址的数据存储在如由控制器的逻辑到物理地址表 163 保存的对应物理地址中。

在典型的存储器系统中，提供比必需的多几个的额外区块的存储容量，以存储地址空间 161 内的数据的量。可提供这些额外区块中的一者或一者以上作为冗余区块，用于代替在存储器的使用寿命期间可能变得有缺陷的其它区块。包含在个别元区块内的区块的逻辑分组通常可出于各种原因而改变，所述原因包含冗余区块替代原本指定给元区块的有缺陷区块。一个或一个以上额外区块（例如元区块 171）通常保存在擦除区块集区中。当主机将数据写入到存储器系统时，控制器将由主机指派的逻辑地址转换成擦除区块集区中的元区块内的物理地址。其它没有被用于将数据存储在逻辑地址空间 161 内的元区块接着被擦除，且被规定为擦除集区区块，以供在随后的数据写入操作期间使用。在优选形式中，将逻辑地址空间分成多个逻辑群组，每个逻辑群组含有等于物理存储器元区块的存储容量的数据量，因此允许所述逻辑群组一对一地映射到所述元区块中。

当原先存储的数据变得废弃时，存储在特定主机逻辑地址处的数据由新数据频繁地重写。作为响应，存储器系统控制器将新数据写入擦除的区块中，且接着针对那些逻辑地址改变逻辑到物理地址表，以识别那些逻辑地址处的数据所存储到的新的物理区块。接着擦除含有那些逻辑地址处的原先数据的区块，并使其可用于存储新的数据。如果在写入开始时，来自擦除区块集区的经预擦除的区块中没有足够的存储容量，那么此类擦除通常必须在当前数据写入操作可完成之前发生。这可能不利地影响系统数据编程速度。存储器控制器通常仅在主机将新数据写入到其给定逻辑地址时才了解到主机已经使同一逻辑地址处的数据变得废弃。因此，存储器的许多区块可能暂时存储此类无效数据。

区块和元区块的大小逐渐增加，以便高效地使用集成电路存储器芯片的面积。这导

致较大比例的个别数据写入存储小于元区块的存储容量（且在许多情况下，甚至小于区块的存储容量）的数据量。由于存储器系统控制器通常将新数据引导到擦除集区元区块，所以这可能导致元区块的多个部分变得未填满。如果新数据是存储在另一元区块中的某数据的更新，那么来自所述另一元区块的具有与新数据元页的那些逻辑地址邻接的逻辑地址的剩余有效数据元页也合意地以逻辑地址次序复制到新的元区块中。旧的元区块可保留其它有效数据元页。随着时间的过去，这导致个别元区块的某些元页的数据变得废弃和无效，且由被写入到不同元区块的具有同一逻辑地址的新数据代替。

为了保存足够的物理存储器空间以在整个逻辑地址空间 161 上存储数据，周期性地对此类数据进行压缩或合并（垃圾收集）。尽可能地将数据扇区以与其逻辑地址相同的次序保存在元区块内也是合乎要求的，因为这使得以连续逻辑地址读取数据更高效。因此，通常为了此额外目标而执行数据压缩和垃圾收集。第 6,763,424 号美国专利中描述当接收部分区块数据更新时管理存储器的一些方面以及元区块的使用。

数据压缩通常涉及从元区块读取所有有效数据元页，并将所述元页写入到新的元区块，在所述过程中忽略具有无效数据的元页。具有有效数据的元页还优选以与存储在其中的数据的逻辑地址次序匹配的物理地址次序布置。新的元区块中被占据的元页的数目将小于旧的元区块中被占据的元页的数目，因为含有无效数据的元页没有被复制到新的元区块。接着擦除旧的区块，并使其可用于存储新的数据。通过合并获得的额外元页容量可接着用于存储其它数据。

在垃圾收集期间，从两个或两个以上元区块收集具有邻接或几乎邻接的逻辑地址的有效数据的元页，并将其重写到另一元区块中，通常是擦除区块集区中的一个元区块。当所有的有效数据元页都从原先的两个或两个以上元区块复制出来时，可擦除所述元区块以供将来使用。

数据合并和垃圾收集会花费时间，且可能影响存储器系统的性能，尤其是在数据合并或垃圾收集需要在可执行来自主机的命令之前发生时。此类操作通常由存储器系统控制器调度以尽可能在后台发生，但执行这些操作的需要可能导致控制器不得不给主机忙碌状态信号，直到这个操作完成为止。主机命令的执行可延迟的实例是擦除区块集区中没有足够的预擦除元区块用来存储主机想要写入到存储器中的所有数据，且首先需要数据合并或垃圾收集来清除一个或一个以上有效数据的元区块，所述元区块接着可被擦除。因此，注意力已经被引导到管理对存储器的控制，以便使此类中断减到最小。在以下美国专利申请案中描述许多此类技术：2003 年 12 月 30 日申请的题为“Management of

Non-Volatile Memory Systems Having Large Erase Blocks” 的第 10/749,831 号美国专利申请案, 现在的第 2005/0144358 A1 号公开案; 2003 年 12 月 30 日申请的题为“Non-Volatile Memory and Method with Block Management System” 的第 10/750,155 号美国专利申请案; 2004 年 8 月 13 日申请的题为“Non-Volatile Memory and Method with Memory Planes Alignment” 的第 10/917,888 号美国专利申请案, 现在是第 2005/0141313 A1 号公开案; 2004 年 8 月 13 日申请的第 10/917,867 号美国专利申请案, 现在是第 2005/0141312 A1 号公开案; 2004 年 8 月 13 日申请的题为“Non-Volatile Memory and Method with Phased Program Failure Handling” 的第 10/917,889 号美国专利申请案, 现在是第 2005/0166087 A1 号公开案; 2004 年 8 月 13 日申请的题为“Non-Volatile Memory and Method with Control Data Management” 的第 10/917,725 号美国专利申请案, 现在是第 2005/0144365 A1 号公开案; 2005 年 7 月 27 日申请的题为“Non-Volatile Memory and Method with Multi-Stream Update Tracking” 的第 11/192,220 号美国专利申请案; 2005 年 7 月 27 日申请的题为“Non-Volatile Memory and Method with Improved Indexing for Scratch Pad and Update Blocks” 的第 11/192,386 号美国专利申请案; 以及 2005 年 7 月 27 日申请的题为“Non-Volatile Memory and Method with Multi-Stream Updating” 的第 11/191,686 号美国专利申请案。

高效地控制具有非常大的擦除区块的存储器阵列的操作的一个难题是使给定写入操作期间存储的数据扇区的数目与存储器的区块的容量和边界匹配并对准。一种手段是配置元区块, 所述元区块用于以少于最大数目的区块存储来自主机的新数据, 所述数目的区块是存储少于填充整个元区块的量的数量的数据所必需的。2003 年 12 月 30 日申请的题为“Adaptive Metablocks” 的第 10/749,189 号美国专利申请案(现在是第 2005/0144357 A1 号公开案) 中描述适应性元区块的使用。2004 年 5 月 7 日申请的第 10/841,118 号专利申请案(现在是第 2005/0144363 A1 号公开案) 以及 2004 年 12 月 16 日申请的题为“Data Run Programming” 的第 11/016,271 号专利申请案(现在是第 2005/0144367 A1 号公开案) 中描述数据区块之间的边界与元区块之间的物理边界的配合。

存储器控制器还可使用来自 FAT 表(其由主机存储在非易失性存储器中) 的数据, 以更高效地操作存储器系统。一种此类使用是通过解除分配其逻辑地址来了解数据何时已经由主机识别为废弃的。知道这一点将允许存储器控制器在其会正常地通过主机将新数据写入到那些逻辑地址而了解它之前调度含有此类无效数据的区块的擦除。2004 年 7 月 21 日申请的题为“Method and Apparatus for Maintaining Data on Non-Volatile Memory

Systems”的第 10/897,049 号美国专利申请案（现在是第 2006/0020744 A1 号公开案）中描述这种情况。其它技术包含监视将新数据写入到存储器的主机模式，以便推导出给定写入操作是单个文件还是（如果有多个文件）文件之间的边界所在之处。2004 年 12 月 23 日申请的题为“FAT Analysis for Optimized Sequential Cluster Management”的第 11/022,369 号美国专利申请案（现在是第 2006/0020745 A1 号公开案）描述这种类型的技术的使用。

为了高效地操作存储器系统，对控制器来说合乎需要的是尽可能多地知道由主机指派给其个别文件的数据的逻辑地址。数据文件接着可由控制器存储在单个元区块或元区块群组内，而不是当不知道文件边界时分散在更大数目的元区块之间。结果是数据合并和垃圾收集操作的数目和复杂性减小。因此，存储器系统的性能改进。但如上文所述，当主机/存储器接口包含逻辑地址空间 161（图 7）时，存储器控制器难以知道关于主机数据文件结构的较多信息。

参看图 8，以不同的方式来说明图 7 中已经展示的典型逻辑地址主机/存储器接口。主机向主机产生的数据文件分配逻辑地址。存储器系统于是看到这些逻辑地址，并将其映射到实际存储数据的存储器单元的区块的物理地址中。

基于文件的存储器接口与操作

主机与用于存储大量数据的存储器系统之间的一种不同类型的接口不使用逻辑地址空间。主机改为通过唯一文件 ID（或其它唯一参考）和所述文件内的数据单位（例如字节）的偏移地址在逻辑上寻址每个文件。此文件地址是直接给存储器系统控制器的，存储器系统控制器接着保存其自己的关于每个主机文件的数据物理上存储在何处的表。可用如上文相对于图 2 到图 6 所述的同一存储器系统来实施这种新的接口。与上文所述内容的主要不同之处是存储器系统与主机系统通信的方式。

图 9 中说明这种基于文件的接口，其应与图 7 的逻辑地址接口相比较。文件 1、2 和 3 中的每一者的识别以及图 9 的文件内的数据的偏移量被直接传递到存储器控制器。此逻辑地址信息接着由存储器控制器功能 173 转换成存储器 165 的元区块和元页的物理地址。文件目录记住每个所存储的扇区、页或其它文件数据单位所属的主机文件。

图 10 还说明基于文件的接口，其应与图 8 的逻辑地址接口相比较。图 8 的逻辑地址空间和主机保存的 FAT 表在图 10 中不存在。而是通过文件编号和所述文件内的数据的偏移量向存储器系统识别由主机产生的数据文件。存储器系统控制器接着直接将所述文件映射到存储器单元阵列的物理区块，并保存里面存储有主机文件的存储器区块的文件目

录和索引表信息。于是，主机没有必要保存当前对管理逻辑地址接口来说是必需的文件分配表（FAT）。

图 11 是直接数据文件系统的主要功能的概括方框图，所述功能由其处理器且由控制器的其它电路执行的存储器系统固件实行。图 11 提供了可在其中考虑下文描述的特定存储器操作的总体框架。基于文件的接口 601 层在存储器系统与外部主机系统或正在同一存储卡或快闪驱动器上执行的主机应用程序之间传递命令和数据，以实现存储器系统的三个主要功能，即写入文件、删除文件和读取文件。数据存储 in 快闪存储器阵列 603 中。

文件到区块映射功能 605 根据由其识别数据的文件而组织数据在快闪存储器中的存储。对于每个文件，含有所述文件的数据以及其它文件的数据的快闪区块的数目是受限的。这使必须被重新定位以回收文件被删除或修改时所创建的废弃数据空间的不相关文件数据的量减到最小，从而导致改进的性能和存储器持久性。存储器 603 的存储器单元的物理区块是数据管理的基本单位。

数据缓冲和编程功能 607 控制来自文件接口或快闪存储器中的位置的文件数据临时存储在缓冲存储器中、所述文件数据转移到快闪存储器和所述文件数据编程到文件的活跃区块或临时交换区块中。每个文件数据群组的开头优选直接与快闪存储器中的元页的开头对齐。在每个数据群组的偏移地址空间内界定逻辑元页结构。

图 11 的功能 609 控制对存储器 603 的用以读取存储在其中的数据的存取。当由主机命令时，删除文件的数据致使功能 611 更新功能 613 所保存的文件索引信息和功能 615 中的区块记录。

文件数据索引 613 通过唯一文件识别符和所述文件内的数据的偏移地址将存储在存储器 603 中的个别文件编入索引。每个文件的数据被存储为一组具有邻接逻辑偏移地址的数据群组。文件目录识别个别文件在多组数据群组条目的文件索引表（FIT）中的位置。被擦除的（部分地编程有文件数据或含有文件数据以及废弃数据）的区块的身份由区块记录功能 615 保存。

上文所述的垃圾收集和数据合并功能的主要目的是回收未经使用的存储器空间，以用于存储额外数据。在垃圾收集中，将源区块的有效数据从也含有废弃数据的区块复制到具有至少一些经擦除空间的一个或一个以上目的地区块中。这将有效数据收集到数目较少的区块中，从而一旦原先源区块被擦除，便释放由废弃数据占据的容量。在数据合并中，一个部分填充的区块（其因此也含有经擦除但未经使用的空间）的有效数据与另一部分填充的区块的有效数据进行组合。部分填充的区块最常见是由写入已关闭但其最

后一个擦除区块仅部分地填充的新文件引起的。一旦数据被合并，接着擦除含有刚被复制的数据（其于是复制数据）的源区块，并使其可用于存储新数据。

垃圾收集和数据合并两者在本文可一起作为区块回收来处置。功能 617 通过控制有效文件数据从具有未经编程的元页或含有废弃数据的物理区块复制到其它区块来回收区块。这允许原先区块被擦除，以回收其所含有的未经使用的空间，并使此空间可用于存储新的文件数据。功能 619 根据可回收容量的量和经擦除区块的数目，适应性地控制区块回收操作的发生和持续时间。以维持存储器系统的良好总体性能的方式，以相对于新文件数据的写入速率的最佳速率执行区块回收。

在图 11 的功能图中，转换层 621 和接口层 623 在文件接口 601 之上，文件接口 601 与快闪存储器的后端系统介接，并控制其操作。在此实例中，接口层 623 具有根据三个不同协议中的一者，用主机或以其它方式将数据传送到存储器系统外部的能力。文件接口 625 是本文主要描述的文件接口，其中个别文件的数据由唯一文件识别符和所述文件内的逻辑偏移地址识别。对象接口 627 的主要用途是在电子装置之间传递数据文件，其中文件的大小通常是已知的。现存的用于接口 627 的协议包含来自微软公司的媒体传送协议（MTP）和图片传递协议（PTP）。此实例中还包含向后兼容逻辑（LBA）接口 629。以快闪存储卡当前所使用的协议，通过接口 629 传递数据，这与磁盘驱动器系统的情况类似，其中主机将数据寻址到存储器系统的已界定的逻辑地址空间。

转换层 621 包含协议适配器 631、633 和 635，其用于将相应接口协议 625、627 和 629 的协议转换成文件接口 601 的共用协议。命令、数据格式等等通过转换层在不同协议之间转换。LBA 协议适配器 635 另外将存储器系统的逻辑地址空间分成静态文件。这些文件接着由文件接口 601 以与通过接口 625 和 627 传送的不同文件相同的方式进行处置。可参考 2005 年 8 月 3 日申请的发明人为 S. A. Gorobets 的第 11/196,869 号美国专利申请案获得 LBA 协议适配器 635 的功能的细节。在 2005 年 12 月 21 日申请的发明人为 Alan Sinclair 的第 11/316,577 号美国专利申请案中给出转换和接口层 621 和 623 的更多信息。

当新的数据文件被编程到存储器中时，数据被写入到存储器单元的经擦除区块中，从所述区块中的第一物理位置开始，并依次循序地进行完所述区块的位置。数据以从主机接收到的次序被编程，不管所述数据在文件内的偏移量的次序如何。编程继续进行，直到所述文件的所有数据都已经写入到存储器中为止。如果文件中的数据量超过单个存储器区块的容量，那么当第一区块满时，编程在第二经擦除区块中继续进行。以从第

一位置开始的次序，以与第一存储器区块相同的方式对第二存储器区块进行编程，直到文件的所有数据都被存储或第二区块满为止。可以文件的任何剩余数据对第三或额外区块进行编程。存储单个文件的数据的多个区块或元区块无需是物理上或逻辑上邻接的。为了便于阐释，除非另外规定，否则希望本文所使用的术语“区块”指代擦除区块单位或多区块“元区块”，视特定系统中是否使用元区块而定。

图 12 的图说明图 11 中所示的存储器操作的总体运行。个别存储器区块可被视为处于三个状态中的一者。这些是：经擦除区块 641、区块 643，其存储有效文件数据而不具有可回收容量；区块 645，其可含有一些有效文件数据，但还具有来自未经编程的经擦除页和/或存储在其中的废弃（无效）数据的可回收容量。数据通过功能 647 写入到经擦除存储器区块，从而形成类别 643 或 645 中的区块，视所得的经编程区块是否保留任何可回收容量而定。当文件被删除（如由功能 649 所指示）时，含有文件的数据的区块 643 转换成具有可回收容量的区块 645。当在功能 650 中将数据从可回收区块复制到其它区块之后，区块 645 的未经使用的存储容量由功能 651 回收，这导致将这些区块返回到新的数据可写入到的经擦除区块 641 的状态。

参看图 13A，说明将数据文件写入到存储器系统。在此实例中，数据文件 181 大于存储器系统的一个区块或元区块 183 的存储容量，其被展示为在垂直实线之间延伸。数据文件 181 的一部分 184 因此也被写入到第二区块 185 中。将这些存储器单元区块展示为物理上邻接，但它们无需如此。来自文件 181 的数据以与其从主机被串流的方式相同的方式被写入，直到文件的所有数据都已经写入到存储器中为止。在图 13A 的实例中，数据 181 是文件的初始数据。

存储器系统管理和记住所存储数据的优选方式是使用可变大小的数据群组。即，将文件的数据存储为多个数据群组，所述数据群组可以界定的次序链接在一起以形成完整的文件。然而，优选的是，存储器系统控制器通过使用文件索引表（FIT）来保存数据群组在文件内的次序。当来自主机的数据流被写入时，只要文件数据的逻辑偏移地址中或数据将存储在其中的物理空间中存在不连续性，就开始新的数据群组。此物理不连续性的实例是当文件的数据填充一个区块并开始被写入到另一区块中的时候。这在图 13A 中说明，其中第一数据群组填充第一区块 183，文件的剩余部分 184 存储在第二区块 185 中，作为第二数据群组。第一数据群组可由(F0,D0)表示，其中 F0 是数据文件的开头的逻辑偏移量，且 D0 是存储器内文件开始的物理位置。第二数据群组表示为(F1,D1)，其中 F1 是存储在第二区块 185 的开头处的数据的逻辑文件偏移，且 D1 是所述数据所存储

的物理位置。

通过主机-存储器接口传递的数据的量可用若干数据字节、若干数据扇区或以某一其它颗粒度来表达。在通过当前逻辑地址接口与大容量存储器系统通信时，主机最常用字节颗粒度来界定其文件的数据，但接着将字节分组为每者具有 512 个字节的扇区，或分组成每者具有多个扇区的群集。这样做通常是为了简化存储器系统的操作。尽管本文所述的基于文件的主机-存储器接口可使用某一其它数据单位，但原先主机文件字节颗粒度通常是优选的。即，数据偏移量、长度及类似物优选为字节（数据的最小合理单位）而不是用扇区、群集或类似物来表达。这允许用本文所述的技术更高效地使用快闪存储器存储设备的容量。

以图 13A 中所说明的方式写入到存储器中的新文件接着在 FIT 中以所述次序表示为数据群组的索引条目(F0,D0)、(F1,D1)的序列。即，只要主机系统想要存取特定文件，主机就将其文件 ID 或其它标识发送到存储器系统，存储器系统接着存取其 FIT 以识别组成所述文件的数据群组。为了方便存储器系统的操作，个别数据群组的长度<长度>也可包含在其个别条目中。在使用时，存储器控制器计算并存储数据群组的长度。

只要主机以打开状态保存图 13A 的文件，物理写入指针 P 就也优选被保存，以界定用于针对所述文件写入从主机接收到的任何进一步数据的位置。在物理存储器中的文件的结尾处写入用于所述文件的任何新的数据，而不管所述新数据在文件内的逻辑位置如何。存储器系统允许多个文件同时保持打开，例如 4 个或 5 个此类文件，并保存所述文件中的每一者的写入指针 P。不同文件的写入指针指向不同存储器区块中的位置。如果当存储器系统对打开文件的数目的限制已经存在时主机系统想要打开新的文件，那么首先关闭打开的文件中的一者，且接着打开新的文件。

图 13B 说明通过主机将数据附加到图 13A 的先前写入但仍打开的文件的结尾。展示数据 187 由主机系统添加到文件的结尾，其还在所述文件的数据的结尾处写入第二区块 185 中。所附加的数据变成数据群组(F1,D1)的一部分，所述数据群组因此现在含有更多数据，因为现存数据群组 184 与所附加数据 189 之间不存在逻辑或物理地址不连续性。因此，完整的文件在 FIT 中仍表示为索引条目(F0,D0)、(F1,D1)的序列。指针 P 的地址也改变成所存储的附加数据的结尾的地址。

图 13C 中展示将数据区块 191 插入到图 13A 的先前写入的文件中的实例。尽管主机正将数据 191 插入到文件中，但存储器系统将所插入的数据附加在先前写入的文件数据的结尾处的位置 193 处。当数据被插入到打开文件中时，没有必要以其逻辑次序来重写

文件的数据，尽管这可以稍后在主机关闭所述文件之后在后台进行。由于插入的数据完全存储在第二存储器区块 185 内，所以如果形成单个新群组(F1,D3)。但进行此插入导致图 13A 的先前数据群组(F0,D0)被分成两个群组，一个(F0,D0)在插入之前，且一个(F2,D1)在插入之后。这是因为每当存在数据的逻辑不连续性时就需要形成新的数据群组，所述不连续性例如在插入的开头 F1 处且在插入的结尾 F2 处发生。群组(F3,D2)是物理地址 D2 为第二区块 185 的开头的结果。群组(F1,D3)和(F3,D2)即使存储在同一存储器区块中也是分别保存，因为存储在所述群组中的数据偏移量中存在不连续性。具有插入的原先文件于是在存储器系统 FIT 中由数据群组索引条目(F0,D0)、(F1,D3)、(F2,D1)、(F3,D2)以所述次序表示。从图 13A、图 13B 和图 13C 的实例应注意，可在不使存储器中的任何数据废弃的情况下，写入新的或现存文件的新数据。

作为图 13C 中所说明的将数据插入到现存文件中的替代方案，只要数据已经插入，主机就可将文件作为单独文件重写到存储器中。接着，存储器系统可将此单独文件视为新的文件。旧的文件接着被主机删除，且存储器系统可通过回收旧文件存储在其中的空间来作出响应，所述旧文件的数据现在是废弃的。

图 13D 说明另一实例，其中原先以图 13A 中所示方式写入的数据的某一部分被更新。展示数据文件的一部分 195 被更新。不是用所述更新重写存储器系统中的整个文件，而是将文件的经更新部分 197 附加到先前写入的数据。先前写入的数据的部分 199 现在是废弃的。在更新之后，文件在存储器系统 FIT 中由数据群组索引条目(F0,D0)、(F1,D3)、(F2,D1)、(F3,D2)以所述次序表示。图 13A 的所述单个数据群组(F0,D0)再次被分成图 13D 中的片段，在经更新部分之前的一个片段、经更新部分以及在经更新部分之后的一个片段。回收由废弃数据占据的空间 199 是合乎需要的，但优选稍后完成此步骤，而不是作为将文件数据写入到存储器中的过程的一部分。此回收通常将导致被存储的特定文件的数据的更少数目的数据群组。

为了进一步说明可变长度数据群组的使用，图 14A 到图 14E 依次展示涉及同一文件的几个写入操作的序列。首先将原先文件数据 W1 写入到存储器系统的两个区块中，如图 14A 中所示。接着由两个数据群组来界定文件，第一群组在物理存储器区块的开头开始，且要求第二群组在物理存储器区块边界之后。接着通过数据群组的索引条目的以下序列：(F0,D0)，(F1,D1)来描述图 14A 的文件。

在图 14B 中，主机致使图 14A 中所写入的文件数据被更新。经更新的文件数据 U1 紧接在先前群组(F1,D1)之后写入，其中经更新的数据的先前版本变成废弃。图 14A 的先

前群组(F0,D0)缩短成图 14B 的经修订的群组(F0,D0),且先前群组(F1,D1)缩短成群组(F4,D2)。在两个群组(F2,D3)和(F3,D4)中写入经更新的数据,因为它们与存储器区块的边界重叠。所述数据中的一些数据存储在第一存储器区块中。现通过数据群组的索引条目的以下序列:(F0,D0)、(F2,D3)、(F3,D4)、(F4,D2)来描述文件。

通过主机致使新的文件数据 I1 插入来在图 14C 中进一步修改图 14B 的文件。新的数据 I1 紧接在图 14B 的先前群组(F4,D2)之后写入到存储器中,作为图 14C 的新的群组(F5,D6)和(F6,D7),因为所插入的数据与存储器区块的边界重叠。使用第四存储器区块。将图 14B 的先前群组(F0,D0)分成图 14C 中的缩短的群组(F0,D0)和(F7,D5),因为插入了新的数据 I1。现通过数据群组的索引条目的以下序列:(F0,D0)、(F5,D6)、(F6,D7)、(F7,D5)、(F8,D3)、(F9,D4)、(F10,D2)来描述文件。

图 14D 展示图 14C 的数据文件的进一步修改,其将新的数据 W2 附加到文件结尾。新的数据 W2 紧接在图 14C 的先前群组(F10,D2)之后写入,作为图 14D 的新的群组(F11,D8)。现通过数据群组的索引条目的以下序列:(F0,D0)、(F5,D6)、(F6,D7)、(F7,D5)、(F8,D3)、(F9,D4)、(F10,D2)、(F11,D8)来描述文件。

图 14E 中展示对打开文件的第二次更新,其中将经更新的文件数据 U2 写入到图 14D 的文件。经更新的数据 U2 紧接在图 14D 的先前群组(F11,D8)之后在图 14E 中写入,其中所述数据的先前版本变成废弃。图 14D 的先前群组(F9,D4)缩短成图 14E 中的经修订的群组(F9,D4),先前群组(F10,D2)变成完全废弃,且先前群组(F11,D8)缩短以形成新的群组(F14,D9)。在图 14E 的新的群组(F12,D10)和(F13,D11)中写入经更新的数据,从而与区块边界重叠。现需要第五区块来存储文件。现通过数据群组的索引条目的以下序列:(F0,D0)、(F5,D6)、(F6,D7)、(F7,D5)、(F8,D3)、(F9,D4)、(F12,D10)、(F13,D11)、(F14,D9)来描述文件。

根据前面的描述内容,在文件的创建或修改之后,每个文件的数据的偏移量优选以正确的逻辑次序保持连续。因此,举例来说,作为将数据插入到文件中的操作的一部分,由主机提供的所插入数据的偏移量从紧接在插入物之前的偏移量开始是连续的,且在所述插入物之后的已经在文件中的数据递增所插入数据的量。更新现存文件最常见的是导致现存文件的给定地址范围内的数据由类似量的经更新数据代替,因此文件的其它数据的偏移量通常不需要被代替。

将注意,上文相对于图 13 和图 14 所描述的所有数据分配和索引功能都由存储器系统的控制器执行。连同适当的命令一起,主机仅传送被发送到存储器系统的文件 ID 和文

件内的数据的偏移。存储器系统完成剩余的工作。

以刚才描述的方式直接将文件数据从主机写入到快闪存储器中的优势在于这样存储的数据的颗粒度或分辨率可维持与主机的相同。举例来说，如果主机应用程序写入具有 1 字节颗粒度的文件数据，那么所述数据可也以 1 字节颗粒度写入到快闪存储器中。接着以字节数目来测量数据在个别数据群组内的量和位置。即，可在主机应用程序文件内分别寻址的同一数据偏移单位在存储在快闪存储器中时也可在所述文件内分别寻址。接着，区块内的同一文件的数据群组之间的任何边界在索引表中被指定为最近字节或其它主机偏移单位。类似地，区块内的不同文件的数据群组之间的边界以主机偏移量的单位界定。

对于较大区块存储器，本文使用术语“扇区”以表示 ECC 与之相关联的存储数据的单位。因此，当此误差校正代码由存储器系统的控制器产生并与数据一起存储时，扇区是转移到快闪存储器和从快闪存储器转移的数据的最小单位。使用“页”来表示区块内的存储器单元的单位，且“页”是最小编程单位。使用“元页”来表示具有元区块的完整并行性的页。元页是最大编程单位。

根据图 14B 和图 14E 将注意，更新命令导致存储文件所必需的物理空间大于文件中的数据的数据的量。这是因为已经由更新物代替的数据保持存储在存储器中。因此非常希望通过去除废弃的无效数据来将文件的数据合并（垃圾收集）到较小的物理存储空间。因此，更多的存储空间变成可用于其它数据。

还可注意，除了图 14B 和图 14E 的文件数据更新之外，图 14C 的数据插入导致文件数据无次序地存储。即，更新物和插入物在其形成时添加到存储在存储器中的文件的结尾，而它们几乎总是逻辑上定位在文件内的某处。这是图 14B、图 14C 和图 14E 的实例的情况。因此，可能希望对存储在存储器中的文件的数据重新排序，以与文件内的偏移量的次序匹配。这接着改进读取所存储的数据的速度，因为依次读取页和区块将以其偏移量次序来给出文件的数据。这还提供文件的最大可能解分段。但对于存储器系统的性能来说，对文件数据重新排序以使读取更高效不如文件数据合并重要，文件数据合并潜在地释放一个或一个以上存储器区块以用于存储其它数据。因此，对文件中的数据重新排序通常将不是独自完成的，其中益处不值得所添加的操作额外开销，但可在具有少量或无添加的操作额外开销的情况下，作为许多垃圾收集操作的一部分而完成。

由于两个数据更新 U1 和 U2 已经形成的缘故，图 14E 的文件包含存储在存储器中的废弃数据群组（灰暗部分）。因此，用于存储所述文件的存储器容量的量实质上大于文件

的大小，如从图 14E 可明显看出。因此，垃圾收集是适当的。图 15 提供图 14E 的数据文件的垃圾收集的结果的说明。所述文件在垃圾收集之前占据几乎五个区块的存储容量(图 14E)，而同一文件在垃圾收集之后配合在略多于三个的存储器单元区块内(图 15)。作为垃圾收集操作的一部分，将数据从其初始被写入的区块复制到其它经擦除区块中，且接着擦除原先区块。如果对整个文件进行垃圾收集，那么可以与所述文件内的数据逻辑偏移次序相同的物理次序，将所述文件的数据复制到新的区块中。更新物 U1 和 U2 以及插入物 I1 (例如) 在垃圾收集(图 15)之后，以与其在主机文件中出现的次序相同的次序存储。

以文件为基础的垃圾收集通常还导致在正合并的文件内形成新的且不同的数据群组。在图 15 的情况下，通过新数据群组的索引条目的以下新序列：(F0,D12)、(F1,D13)、(F2,D14)、(F3,D15)来描述文件。数据群组的这个数目远小于图 14E 中所示的文件的状态下存在的数据群组的数目。现针对文件的数据已经复制到其中的存储器单元区块中的每一者，存在一个数据群组。作为垃圾收集操作的一部分，文件索引表(FIT)经更新以反映形成所述文件的新的数据群组。

将注意，从图 14E 的数据分配移动到图 15 的数据分配涉及相当大量的数据复制。从图 14E 中的区块中的每一者读取有效数据，且接着将其写入到图 15 的四个经擦除区块中。尽管这大大简化了供将来使用的文件的数据群组，但复制所述数据要花费相当大量的时间。因此，不是在文件基础上对所述数据进行垃圾收集，替代方案是在区块基础上进行垃圾收集，即使在数据作为文件直接存储在存储器中时也是如此。可通过对那些具有最少量的要复制到另一区块的数据的区块执行垃圾收集来回收区块的未经使用的容量。在上文参考的 2006 年 5 月 8 日申请的题为“Reclaiming Data Storage Capacity in Flash Memories”的申请号为 11/382,232 的专利申请案中描述这种情况。

因此，当处于图 14E 的状态时，回收保存文件的数据的区块个别地对区块而不是对存储同一文件的数据的多个区块进行操作。举例来说，如果图 14E 的包含存储器系统的任何区块的最少量的有效数据的第二区块 002 在给定时间被考虑用于回收操作，那么其单个数据群组将被从一个区块复制到另一个(经擦除区块 010)，如图 16 中所示。新的区块于是含有单个数据群组(F8,D16)，且区块的剩余部分是经擦除容量，可将新的数据写入到所述经擦除容量中。已经从图 14E 中的里面存储有数据的区块回收了所述经擦除容量。接着通过组成所述文件的数据群组的索引条目的以下序列：(F0,D0)、(F5,D6)、(F6,D7)、(F7,D5)、(F8,D16)、(F9,D4)、(F12,D10)、(F13,D11)、(F14,D9)来描述文件。根据区块回

收过程，图 14E 中展示的有关块保持不变，直到它们个别地满足回收操作的标准为止。

文件数据索引

图 17 说明在若干不同时间 0、2、4、X 和 Y 中的每一者时一个文件的文件索引表(FIT)中的索引条目的序列。这些序列分别是上文相对于图 14A、图 14C、图 14E、图 15 和图 16 所描述的序列。FIT 中的数据优选在没有主机系统的辅助的情况下，通过存储器控制器写入并保持最新。主机系统在将数据写入到存储器系统时供应路径名、文件名和文件内的数据的偏移量，但主机不参与界定数据群组或其存储在存储器单元阵列中何处。在图 17 的条目中，图 14A 到图 14E 的存储器单元区块从左边以 1 开始编号。因此，举例来说，对于处于图 14C 中所说明的状态的文件来说，其第三数据群组(F6,D7)在图 17 中被注释为存储在区块 004（从左边开始数的第四个区块）中，距所述区块的初始地址 D7 个字节。每个数据群组的长度优选也包含在所述表的每个条目中。

在对文件的改变导致对所述文件的数据群组的修改时，或以其它较不频繁的时间间隔，存储器控制器对一个文件的图 17 中所示的所述序列索引条目进行重写。控制器可将此类改变存储在其存储器中，且接着一次将所述改变中的许多改变写入到快闪存储器。在任一时刻，针对一个文件仅存在一组有效索引；图 17 中展示五组此类索引，其在不同时间界定所述文件。时间 X 和 Y 时的索引展示两种不同类型的回收操作的结果，第一种是在文件基础上的，且另一种是在区块基础上的，如上文所述。存储器系统控制器根据需要使用文件的当前组索引，以将额外数据编程到所述文件、从所述文件读取数据、对里面存储有其数据的文件或区块的数据进行垃圾收集，且潜在地用于其它操作。因此，如果个别文件索引条目按照其文件偏移量（Fx）的次序存储，那么 FIT 更容易使用，但如果不这样，那么控制器无疑可以所述逻辑次序读取所述条目。存储器控制器读取特定文件的索引条目的最常见的原因是在执行主机命令的过程中。

第一特定文件索引实施例

参看图 18，与直接数据文件接口一起操作的存储器系统中所存储的文件由唯一文件 ID 201 识别，所述文件 ID 201 是邻接组中的数值。目录 203 含有存储在存储器装置中的每个有效文件的条目。目录 203 优选具有平坦的分级结构，且不支持子目录。这允许存储器中的直接数据文件系统独立于主机内所使用的操作系统。符合特定主机操作系统协议的分级目录可由直接数据文件系统外部的一层固件支持，从而也支持结合文件元数据（还被称为文件“属性”信息）设施而使用所述平坦目录结构。

对先前已经描述的内容进行总结，文件的数据被存储为一组数据群组，每个所述数

据群组横跨文件偏移地址空间和物理地址空间两者中的一系列邻接地址。文件的所述组内的数据群组彼此不需要具有任何特定物理地址关系。数据群组是具有文件的邻接偏移地址的一组文件数据，以单个存储器区块中的邻接物理地址编程。文件通常将被编程为许多数据群组。数据群组可具有一个字节到一个区块之间的任何长度。文件索引表 (FIT) 205 以偏移地址次序，提供待识别的文件的有效数据群组的位置的记录。文件的一组 FIT 条目被称为 FIT 记录，且由目录 203 中的条目指定。

FIT 205 具有两个组成部分，活动文件索引区块 207 和一个或一个以上非活动文件索引表区块 209。直接数据文件装置内的活动文件的 FIT 记录位于活动文件索引区块 207 中。如下文所述，这由目录 203 间接寻址，这允许在不更新那些文件的目录 203 中的条目的情况下，对活动文件的 FIT 记录作出修改。对于非活动文件，FIT 记录由非活动文件索引区块 209 中的文件目录 203 直接寻址。在任一种情况下，FIT 记录指定组成由唯一文件 ID 201 个别地指定的文件的数据群组 211 的位置。出于交叉参考的目的，每个数据群组可编程有标头，所述标头含有其所属的文件的文件 ID。如图 18 的 213 处所指示，可修改文件的 FIT 记录，例如可能在回收里面存储有文件的数据的区块之后要求这样做。

图 19 中展示图 18 的稍微扩展的图，其中共用组成部分由相同的参考标号识别。展示多个非活动文件索引区块 209。非活动 FIT 编程区块 209a 已经擦除了存储器页，新写入的数据文件的新的 FIT 记录可写入到所述存储器页中。一旦此区块变满，其接着就被指定为非活动文件 FIT 区块 209b，且将新的经擦除区块指定为非活动 FIT 编程区块。针对保存在系统中的每个文件 ID，文件目录 203 通常指向所有的 FIT 区块 207、209a 和 209b 中的 FIT 记录。

特定文件的 FIT 记录是存储在活动文件索引区块 207 中还是存储在非活动文件索引区块 209 的一者中取决于文件是否在不久的将来非常有可能被修改。如果是，那么所述文件被视为活动文件，其 FIT 记录保存在区块 207 中，且如果不是，那么所述文件被视为非活动文件，其 FIT 记录保存在区块 209 的一者中。当存储器系统与向其发送“打开文件”和“关闭文件”命令的主机系统一起操作时，那么所述存储器可将所有的打开的文件视为活动，且将所有关闭的文件视为非活动。使用这些命令的主机通常保存不超过预定数目（例如五个）的若干打开的文件。因此，由主机写入到其打开的文件的的数据的 FIT 记录存储在区块 207 中，而关闭的文件的 FIT 记录存储在区块 209 的一者中。

然而，一些主机不向存储器系统发送打开和关闭文件命令。在这种情况下，存储器

系统通过监视主机将数据写入到文件的行为来确定文件是否在不久的将来非常有可能被修改。如果由主机写入的新的数据群组在存储区块的开头和结尾的中间某处结束，那么可认为最有可能的是主机已经停止将数据写入到所述文件，至少暂时停止。然而，如果新的数据群组在与存储区块的结尾重合处结束，那么可认为更有可能是主机具有更多要写入到文件的数据。在这种情况下，存储器系统最有可能界定数据群组在与存储区块的结尾重合处结束，所以不是主机已经完成了写入所述文件的数据的指示。

直接数据文件接口系统中的文件目录 203 针对所述系统所支持的每个文件 ID 含有一个条目。将所述目录组织为一组对应于装置所支持的循序文件 ID 值的邻接条目。指定文件 ID 的条目可作为特定逻辑页内的特定条目编号直接存取，如图 20 中所示。定位指定文件 ID 的条目不需要目录搜索。

目录条目指定含有文件的数据的索引信息的 FIT 记录的位置，以及组成所述 FIT 记录的 FIT 条目的数目。这在图 20 中分别由字段 217 和 219 展示。如果字段 217 的内容指向非活动文件的记录，那么其含有区块 209 中存储有文件的 FIT 记录的一者中的位置的物理地址。然而，如果文件是活动的，那么字段 217 含有指向保持与经更新或改变的文件 FIT 条目相同的中间位置的逻辑地址，如后文阐释。

目录包括 P 个逻辑页，且每个页包括 E 个目录条目。因此，目录具有 $P * E$ 个文件的容量。目录存储在专用于所述目的的一个或一个以上区块中。如果存在 D 个目录区块，那么每个目录区块含有 P/D 个逻辑页，且具有 $P/D * E$ 个文件的容量。根据所含有的逻辑页的范围，目录区块被指定为 **DIR0** 到 **DIR(D-1)**。目录区块的数目由已经同时存在于装置中的文件的最大数目来确定。如果现存区块不含有待指派的文件 ID 的条目，那么添加目录区块。然而，如果待支持的文件数目减小，那么可压缩所述目录。

图 21 说明可针对允许逻辑页更新的目录区块而利用的示范性物理结构。每个条目（在邻近的垂直虚线之间）含有文件 ID 的 FIT 记录的规格，或指示针对所述文件 ID 不存在文件的空值。当含有文件的数据的数据群组的数目超过 FIT 记录中所允许的 FIT 条目的最大数目时，可针对所述文件分配连续文件 ID，以允许使用连续 FIT 记录。在分配给文件的初级文件 ID 的目录条目内指定连续文件 ID。目录位于专用于所述目的的一个或一个以上区块中。在控制区块的控制日志中的目录区块列表中界定目录区块的物理位置。

图 21 中给出目录条目 225 的说明。其具有以下四个字段：

- 1) FIT 逻辑区块编号 (227)：此字段识别目标文件 ID 的 FIT 记录定位在其中的 FIT

逻辑区块的编号。对应的物理区块地址由目录区块中的 FIT 区块指针中的一者界定，图 21 中说明示范性物理指针 229。字段 227 指向 FIT 区块指针中的一者。FIT 逻辑区块 N 为活动文件索引区块保留。FIT 逻辑区块编号的值 0 表示装置中不存在所述文件。

2) FIT 逻辑页编号 (231): 此字段识别目标文件 ID 的 FIT 记录定位在其内的活动文件索引区块 207 或非活动 FIT 区块 209 (图 19) 中的逻辑页的编号。在 FIT 记录存储在活动文件索引区块 207 (图 19) 中的情况下，其为记录定位在其中的页的间接地址。在非活动 FIT 编程区块 209a 或非活动 FIT 区块 209b 的情况下，其为页的直接物理地址。

3) FIT 页条目编号 (233): 此字段识别目标文件 ID 的 FIT 记录开始处的页内的循序条目编号。

4) FIT 条目数目 (235): 当此字段中的引导位是零时，所述字段指定目标文件 ID 的 FIT 记录内的 FIT 条目的数目。当此字段内的引导位是 1 时，目标文件 ID 的 FIT 条目的数目超过 FIT 记录中所允许的 FIT 条目的最大数目。在这种情况下，所述字段指定待用于所述文件的连续文件 ID。在连续文件 ID 的目录条目中，此字段指定目标和连续文件 ID 的 FIT 记录内的 FIT 条目的总数。

图 21 的 FIT 区块指针 (实例是指针 229) 指定存在于装置中的 FIT 逻辑区块中的每一者。它们只有在目录区块中的最后一个经编程页中才有效。针对可能存在于装置中的每个 FIT 逻辑区块编号存在一个 FIT 区块指针。FIT 区块指针中存在以下三个字段:

1) FIT 区块地址 (237): 此字段指定当前分配给目录条目 225 的字段 227 的 FIT 逻辑区块编号。

2) 有效 FIT 记录的数目 (239): 此字段指定 FIT 逻辑区块中存在的有效文件的数目。

3) 废弃 FIT 条目的数目 (241): 此字段指定 FIT 逻辑区块中存在的废弃 FIT 条目的数目。

目录逻辑页指针 243 (图 21) 充当目录区块的逻辑到物理页映射。针对区块中的每个逻辑页存在一个指针，且所述指针指定所述逻辑页的当前物理页地址。目录页指针条目只有在目录区块中的最近编程的物理页中才有效。

所述目录可以逻辑页为单位而更新。可在单个编程操作中更新逻辑页内的任何数目的条目。页可被读取，一个或一个以上条目可被更新，且所述页可在目录区块中的下一可用经擦除页中重新编程。目录页指针和 FIT 区块指针同时被重新编程，以指定所有字段的当前值。

当目录区块中的最后一个页已经被编程时，压缩所述区块，并将其重写在经擦除区

块中，所述经擦除区块变成新的目录区块。在擦除废弃目录区块之前，通过将目录页指针所界定的所有有效页都复制到经擦除区块来执行压缩。

文件目录所指向的文件索引表（FIT）包含 FIT 条目串，其中每个 FIT 条目识别数据群组的文件偏移地址和在快闪存储器中的物理位置。FIT 优选含有存储在装置中的文件的所有有效数据群组的条目。废弃数据群组不需要由 FIT 索引。以文件偏移地址次序，将文件中的数据群组的一组 FIT 条目保存为连续条目。所述组条目被称为 FIT 记录。FIT 连同活动文件的索引区块保存在一组 FIT 区块中。FIT 区块的数目将视装置中的数据群组的数目而改变。在装置的操作期间，将创建新的 FIT 区块，且去除 FIT 区块。

图 22 中给出 FIT 的逻辑结构的实例。通过文件的目录条目的 FIT 区块地址 237 和 FIT 逻辑页编号 231（图 21）到达特定文件的 FIT 记录 247。个别 FIT 记录包括文件内的数据群组的一组邻接 FIT 条目。所述组中的条目优选是连续的且具有文件偏移地址的次序。

文件的每个 FIT 记录优选包括 FIT 标头 249（图 22）作为 FIT 记录的第一条目。FIT 标头具有等于 FIT 条目的整数数目的固定长度。FIT 标头具有三个字段，如下：

1) 文件 ID：文件 ID 标识目录中的文件的条目。

2) 编程区块：只要 FIT 记录的经更新版本写入 FIT 中，文件的编程区块的当前物理位置就记录在 FIT 标头中。这在文件由主机重新打开时用来定位文件的编程区块。其还可用来使 FIT 记录与已选来进行编程区块合并的文件的编程区块之间的对应性生效。

3) 编程指针：只要 FIT 记录的经更新版本写入 FIT 中，文件的编程区块内的编程指针的当前值就记录在 FIT 标头中。这在文件由主机重新打开时，或在已经针对编程区块合并选择了编程区块时，用来界定用于对文件的编程区块内的数据进行编程的位置。

文件的 FIT 记录通常具有指定组成所述文件的数据群组的物理位置的若干额外条目。其中，针对示范性文件，图 22 中说明一个 FIT 条目 251。个别 FIT 条目具有四个字段，如下：

1) 偏移地址：偏移地址是与数据群组的第一个字节有关的文件内以字节计的偏移量。

2) 长度：这界定数据群组内的文件数据以字节计的长度。完整的数据群组的长度比这个值长数据群组标头的长度。

3) 指针：这是指向数据群组的开头的快闪区块中的位置的指针。所述指针具有以下两个子字段：1) 区块地址，其界定含有数据群组的物理区块，以及 2) 字节地址，其界定数据群组的开头的区块内的字节偏移量。此地址含有数据群组标头。

4) EOF 旗标: EOF 旗标是识别位于数据文件的结尾的数据群组的单个位。

图 23 说明活动文件索引区块 207 (图 18 和图 19) 的物理格式。在专用于活动文件的索引区块内更新活动文件的 FIT 记录。区块的每个页仅存储一个或一个以上条目的单个 FIT 记录, 且使用逻辑 FIT 记录的间接寻址, 以允许在活动文件索引区块中容易地更新 FIT 记录。当逻辑 FIT 记录更新时, 其全部重写在下一可用经擦除页中, 且此最近编程的页中的一组 FIT 页指针提供逻辑页到物理页映射。针对系统中的每个同时活动的文件存在一个逻辑 FIT 记录。

如上文所述, 当目录条目中的对应 FIT 逻辑区块编号 227 与活动文件索引区块有关时, 文件的文件目录条目中的 FIT 逻辑页编号 231 (图 21) 指定逻辑 FIT 记录。逻辑 FIT 记录可在活动文件索引区块中更新和重写, 而不需要更新文件的文件目录条目。FIT 逻辑页编号 231 是 FIT 记录的间接地址, 且通过作为含有经更新 FIT 记录的页的一部分更新和重写的一组 FIT 逻辑页指针 255 中的一者映射到物理页。

活动文件索引区块 207 中的最近编程的页中的一组 FIT 页指针 255 (图 23) 指定逻辑 FIT 记录的有效版本定位在其中的物理页。FIT 页指针的空值表示活动文件索引区块中的 FIT 逻辑页编号尚未分配。针对可同时存在于装置中的每个活动文件存在一个 FIT 页指针字段。

当活动文件索引区块中的最后一个页已经被编程时, 所述区块被压缩并重写在经擦除区块中, 所述经擦除区块变成新的活动文件索引区块。在擦除废弃活动文件编程区块之前, 通过将如由 FIT 页指针界定的所有有效页复制到经擦除区块来执行压缩。

当活动文件通过被关闭或其它方式而变得非活动时, 其 FIT 记录被复制到下文所述的非活动 FIT 编程区块中, 且活动 FIT 区块中的 FIT 记录的数据变成废弃。

非活动 FIT 区块提供存储在存储器系统中的所有数据文件 (而不是当前活动文件) 的 FIT 记录的存储。单个 FIT 编程区块提供用于对 FIT 区块中的 FIT 记录进行编程的构件。参看图 24, 给出非活动文件 FIT 编程区块 209a (图 19) 的细节。当非活动 FIT 编程区块中的所有页都被编程时, 所述区块变成非活动 FIT 区块 209b, 且将经擦除区块分配为新的非活动 FIT 编程区块。

如图 24 中所示, 非活动 FIT 编程区块中的每个 FIT 记录直接从目录寻址。目录条目中的 FIT 逻辑页编号 231 (图 21) 直接指向非活动 FIT 编程区块中的物理页, 且目录条目中的 FIT 页条目编号 233 识别 FIT 记录的开头。多个 FIT 记录可存储在非活动 FIT 编程区块中的单个页中。

优选要求图 24 的非活动 FIT 编程区块中的页内的所有 FIT 记录具有如图 20 中所示的目录的同一逻辑页内的目录条目。这允许非活动 FIT 编程区块页中的所有 FIT 记录的目录条目在目录区块中用单个编程操作进行更新。

当文件通过被打开或其它方式而变成活动，且其 FIT 记录在活动文件索引区块中创建，或文件被删除时，非活动 FIT 编程区块中针对所述文件可能存在的任何 FIT 记录变成废弃。

当图 24 的非活动 FIT 编程区块的最后一个页已经被编程时，创建非活动 FIT 区块 209b（图 19）。

图 25 提供对 FIT 记录进行的操作的实例。一个此类操作 261 导致数据被写入到活动文件。当数据被写入到活动文件时，用于对文件数据进行编程的算法要求不时地更新 FIT。将正被写入的文件的经更新的 FIT 记录编程到活动文件索引区块 207 中。文件的目录条目不需要更新，因为针对活动文件索引区块中的逻辑 FIT 记录使用间接寻址方案，如图 23 中所示且上文所述。

当现存非活动数据文件或新的数据文件打开或以其它方式变成活动（图 25 中展示为功能 263）时，通过将其指针写入活动文件索引区块 207 中的 FIT 页指针字段中（见图 23），来在活动文件索引区块 207 中分配逻辑 FIT 记录。如果所述文件是现存文件，那么其 FIT 记录也从其在 FIT 区块 209b 或 FIT 编程区块 209a 中的现存位置移动到活动文件索引区块 207。可通过单个页编程操作将 FIT 页指针字段和 FIT 记录写入活动文件索引区块 207 中。目录中只有单个页需要更新，以更新已经打开或以其它方式变得活动的文件的目录条目。

图 25 的 265 处指示数据文件的关闭。当活动文件通过被关闭或其它方式变成非活动时，其 FIT 记录从活动文件索引区块 207 移动到 FIT 编程区块 209a。当含有此 FIT 记录的页在 FIT 编程区块 209a 中编程时，尽可能多的与同一目录页有关的 FIT 记录优选从 FIT 区块移动到 FIT 编程区块 209a 的同一页。这为含有废弃 FIT 条目的 FIT 区块提供垃圾收集机制。应根据区块中已经含有的废弃 FIT 条目的数目来选择用于移动此类 FIT 记录的源 FIT 区块。给予具有最高数目的废弃 FIT 条目的 FIT 区块最高优先级。优选不从含有少于预定数目的废弃 FIT 条目的 FIT 区块移动 FIT 记录。

FIT 区块中的废弃 FIT 条目的数目在最后编程的目录页中的 FIT 区块指针字段中界定。目录页应经编程以更新已经移动的 FIT 记录的条目，且更新 FIT 区块指针和目录页指针字段中的参数。活动文件索引区块页应经编程以将空条目插入已经从区块移出的逻辑

辑 FIT 记录的 FIT 页指针中。

垃圾收集操作 267 (图 25) 可与 FIT 区块的垃圾收集或数据文件或数据区块的垃圾收集有关。首先, 论述 FIT 区块 209b 中的一者的垃圾收集。不时地需要对 FIT 区块进行擦除操作, 以恢复由废弃 FIT 条目占据的容量。在擦除 FIT 区块之前, 将所有有效 FIT 记录从 FIT 区块 209b 移动到 FIT 编程区块 209a。上文描述的使文件变得非活动的过程可能不给所有有效 FIT 记录移动的机会。因此, 希望执行 FIT 垃圾收集过程, 以从待擦除的 FIT 区块移动所有有效 FIT 记录。

具有最高数目的废弃 FIT 条目的 FIT 区块优选被指定为 FIT 垃圾收集区块, 且接着应对此区块执行 FIT 垃圾收集操作, 直到所有的有效 FIT 记录都已经移动为止。接着将另一 FIT 区块指定为 FIT 垃圾收集区块。选择 FIT 垃圾收集区块的页以移动到 FIT 编程区块, 且接着用来自与同一目录页有关的 FIT 垃圾收集区块的其它 FIT 记录 (如果可用) 来填充所述页。如果页中剩余有待编程的空间, 那么可进一步用来自其它 FIT 区块的 FIT 记录来填充所述空间。应基于相关目录页中的条目, 来选择待与选定页一起移动的其它 FIT 记录。

目录页优选经编程以更新已经移动的 FIT 记录的条目, 且更新 FIT 区块指针和目录页指针字段中的参数。FIT 垃圾收集操作以所希望的方式调度以与数据垃圾收集操作同时周期性地发生。针对数据垃圾收集的每 N 个页编程操作, 执行 FIT 垃圾收集的一个页编程操作。

现将描述数据文件或数据区块的垃圾收集 267。数据垃圾收集操作可导致需要更新关闭或以其它方式变成非活动的文件的 FIT 记录。这在并入有所需的更新的情况下, 通过将 FIT 记录从 FIT 区块移动到 FIT 编程区块来实现。优选与同一目录页有关的其它 FIT 记录来填充在 FIT 编程区块中编程的页。

第二特定文件索引实施例

数据在由文件 ID 识别的文件对象中写入到直接数据文件存储器系统, 且所述文件由所述文件对象内的逻辑偏移地址寻址。数据被存储为一组数据群组, 每个数据群组具有邻接的逻辑偏移地址和邻接的物理位置。文件索引使文件 ID 和逻辑偏移地址与数据群组的位置相关。

目录直接界定文件对象的索引条目在文件索引标表 (FIT) 中的位置。通常使用直接索引, 但间接索引方案也用于在随后不需要修改文件目录的情况下, 频繁地修改一些文件的索引条目。所述文件索引还支持分别来自文件数据的文件属性信息的存储。

DFS 装置内的文件是具有单个逻辑身份的数据对象。文件由用数字表示的文件 ID 识别，且文件内的数据由偏移地址识别。

DFS 装置响应于来自主机的命令而指派文件 ID 值，以创建文件对象。DFS 系统执行文件索引，以跟踪每个文件对象内的数据的物理位置和每个文件对象的属性的物理位置。

图 26 中展示文件索引的分级结构。文件的目录条目包含三个初级字段、唯一文件 ID 和两个指向存储器内的其它地址的指针。指针 1 指向具有所述文件 ID 的文件在 FIT 中的 FIT 记录，且指针 2 指向所述文件的文件属性记录 (ATR)。目录、FIT 和 ATR 优选存储在快闪存储器中的单独组区块中。

目录针对装置中存在的每个文件对象含有一个条目。所述条目在非重叠的文件 ID 值范围中存储，其中每个范围分配给单独的页。根据文件 ID 值来对范围内的条目进行排序。可通过读取单个页并在所述页内执行二进制搜索来找到目标文件 ID 的目录条目。

文件索引表 (FIT) 直接由目录中的条目寻址，以便使其占据的空间减到最小。其包括记录的集合，每个记录描述组织成页的一个文件对象。通过将页移动到同一或另一 FIT 区块中的未经编程的位置的读取/修改/写入操作来更新 FIT 中的记录的页。FIT 中的页中的记录都与目录中同一页中的条目有关，且 FIT 页因此可被更新，随后只需要修改单个目录页。

使用单独的临时文件索引表 (TFIT) 区块来存储在不久的将来可能要修改的 FIT 记录。文件对象的记录存储在 TFIT 中的逻辑页中，且由目录条目间接寻址。可在不需要更新目录的情况下，修改 TFIT 中的记录。

针对由文件 ID 识别的文件对象，存储文件属性记录 (ATR)。属性记录的内容由主机系统界定，且不由存储器系统解译。一般来说，可允许主机将被认为在主机与存储器系统的操作中重要或有用的文件对象的任何属性存储在 ATR 中。文件属性的索引方案与针对文件索引表所使用的方案相同。

图 27 中展示直接数据文件存储器系统中所使用的一组示范性文件索引结构。所述文件目录 DIR 包含描述由文件 ID 识别的文件对象的条目集合。对于存储器系统中存在的每个文件对象都在 DIR 中存在一个条目。DIR 包含在一个或一个以上专用 DIR 区块中。

每个 DIR 区块含有固定数目的逻辑页，每个逻辑页可通过将其重写到下一可用物理页来更新。含有有效 DIR 条目的页被分配有逻辑页编号。DIR 区块中的逻辑页的数目被指定为区块中的物理页的数目的 25%。

在 DIR 区块的最后一个页已经写入之后，通过将所有有效页写入到经擦除区块并擦

除原先 DIR 区块来压缩所述区块。

DIR 页以其文件 ID 值的次序含有一组 DIR 条目。有效 DIR 条目占据 DIR 页中的一组邻接的条目位置，但不需要填充完整的页。所述组内没有废弃条目，且文件 ID 值无需是邻接的。DIR 页中的文件 ID 值的范围并不与任一其它 DIR 页中的文件 ID 值的范围重叠。

当需要插入新的文件的条目时，从 DIR 索引中的信息识别具有包含新区块的文件 ID 值的文件 ID 范围的 DIR 页。在文件 ID 范围中的适当位置处插入新的条目，且 DIR 页被重写。当必须移除条目时，DIR 页在无所述条目的情况下被压缩且被重写。

当对已经变满的 DIR 页作出添加时，将空闲的逻辑页分配为新的 DIR 页，且将已经变满的 DIR 页的文件 ID 范围分成两个近似相等的非重叠范围，将所述范围写入两个可用 DIR 页中。

当具有邻近文件 ID 范围的两个 DIR 页中的有效条目的总数下降到 DIR 页中的条目位置的数目的 70% 以下时，两个 DIR 页的范围被合并，且写入两个 DIR 页中的一者中。另一个未经使用的页接着变成空闲逻辑页。

图 27 的 DIR 区块的 DIR 条目含有三个字段：

1) 文件 ID

2) 指向 FIT 记录的指针。指针界定 FIT 页逻辑识别符和 FIT 记录的字节偏移量。在特定实例中，FIT 页逻辑识别符识别可由同一 DIR 页中的条目参考的至多达 16 个单独 FIT 页中的一者。所述识别符通过含有所述条目的 DIR 页内的 FIT 索引字段转换成物理区块地址和页编号。字节偏移量识别 FIT 记录在所识别的 FIT 页内的位置。所述指针或者可指向 TFIT。指针内的保留位被设置成表示 FIT 记录驻存在 TFIT 中，且指针界定 TFIT 中的逻辑页。

3) 指向文件属性记录的指针。指针界定 ATR 页逻辑识别符和 ATR 记录的字节偏移量。在特定实例中，ATR 页逻辑识别符识别可由 DIR 页中的条目参考的至多达 16 个单独 ATR 页中的一者。所述识别符通过含有所述条目的 DIR 页内的 FIT 索引字段转换成物理区块地址和页编号。字节偏移量识别 ATR 记录在所识别的 ATR 页内的位置。

单独的有效 FIT 索引字段存在于每个有效 DIR 页中。所述有效 FIT 索引字段用来将 FIT 页的逻辑识别符转换成定位有所述 FIT 页的物理区块地址和页编号。其针对页中的任一条目内用来指向 FIT 记录的每个逻辑识别符含有一个条目。

有效 DIR 索引字段只存在于最近写入的 DIR 页中。所有先前写入的 DIR 页中的字段

中的信息都废弃。其目的是为了支持 DIR 条目的排序和逻辑页到物理页的映射。

DIR 索引针对每个可能的逻辑页含有一个条目，所述条目根据逻辑页编号而排序。每个条目具有三个字段。

1. 逻辑页的分配状态旗标。
2. 与页中的第一个条目有关的文件 ID。这允许建立和高速缓存每个 DIR 页中的文件 ID 值的范围。
3. 指向逻辑页映射到的 DIR 内的物理页的指针。

图 27 的 FIT 包含记录的集合，每个记录针对由文件 ID 识别的文件对象含有控制数据和文件数据索引信息。一个条目存在于装置中所存在的每个文件对象的 FIT 或 TFIT 中。大多数文件对象的 FIT 记录存储在 FIT 中，且 TFIT 含有只针对在不久的将来非常有可能被修改的文件对象的 FIT 记录。

FIT 中的 FIT 记录由 DIR 条目直接寻址，且当 FIT 页修改时，DIR 页必须修改。

FIT 包含在一个或一个以上专用 FIT 区块中。只有一个 FIT 区块含有 FIT 记录可写入到其中的未经编程的页。在由 FIT 写入指针识别的此区块中的下一未经编程的页位置处编程所有 FIT 记录信息。当区块中的最后一个页已经被编程时，FIT 写入指针移动到经擦除区块的第一个页。FIT 区块可含有由已经重写的 FIT 页引起的废弃页，和由已经移除的 DIR 条目引起的有效 FIT 页内的废弃 FIT 记录。可从 FIT 区块回收废弃容量。

FIT 页以与 DIR 页中的条目相同的次序，含有一组由同一 DIR 页内的 DIR 条目参考的 FIT 记录。DIR 页可参考多个 FIT 页。因此，可对 FIT 页进行修改，随后只需要修改单个 DIR 页。可通过读取所述页，接着更新或添加一个或一个以上 FIT 记录来修改 FIT 页。通过压缩所述页来移除任何废弃 FIT 记录，且接着在由 FIT 写入指针识别的位置处编程所述页。

FIT 页标头存储对 FIT 页与其相关联的 DIR 页的参考，以及所述 FIT 页内的 FIT 记录信息的长度。FIT 页标头还可存储在 FIT 页被写入时存在于 FIT 区块的每一者中的废弃页的数目的记录。此信息通常将只在最新写入的 FIT 页标头中有效。

图 27 的 FIT 区块的 FIT 记录含有以下字段：

- 1) 文件 ID
- 2) 文件状态。如上文参考的第 11/382,224 号美国专利申请案中所述，这是从 0 到 9 的数字，每个状态指定允许含有文件的数据的物理区块的类型的唯一预定义的组合。通常，当文件的状态改变时，更新 FIT 记录另外是必需的，因为所述文件的一个或一个以

上数据群组已经改变。

- 3) 文件的活动区块的地址。
- 4) 文件中的数据群组的数目。

文件中的每个数据群组的一个字段，其包括：

- a) 文件内以字节计的逻辑偏移量。
- b) 以字节计的长度。
- c) 指向物理位置的指针。

图 27 的 TFIT 区块用来存储临时的 FIT 记录；即，在不久的将来非常有可能被修改的 FIT 记录。如相对于第一实施例所论述，这可包含（例如）利用打开和关闭文件命令的系统中的打开的文件。TFIT 中的 FIT 记录由 DIR 条目间接寻址，且可在不需要更新 DIR 的情况下修改 TFIT 条目。

TFIT 包含在含有固定数目的逻辑页的单个专用 TFIT 区块中，可通过将每个逻辑页重写到下一可用物理页来更新所述逻辑页。含有有效 TFIT 条目的页被分配有逻辑页编号。TFIT 区块中的逻辑页数目被指定为区块中的物理页的数目的 25%。

在 TFIT 区块的最后一个页已经被写入之后，通过将所有有效页都写入到经擦除区块且擦除原先 TFIT 区块来压缩所述区块。FIT 记录并不同时存在于 FIT 和 TFIT 两者中。

TFIT 页只含有一个 FIT 记录。TFIT 中的 FIT 记录可与上文针对 FIT 区块所述的 FIT 记录相同。有效 TFIT 索引字段只存在于最近写入的 TFIT 页中。所有先前写入的 TFIT 页中的字段中的信息废弃。其目的是将逻辑页映射到物理页。TFIT 索引含有每个可能逻辑页的条目，所述条目根据逻辑页编号排序。每个条目具有两个字段：

1. 逻辑页的分配状态旗标。
2. 指向逻辑页映射到的 TFIT 内的物理页的指针。

图 27 的 ATTR 区块是记录的集合，每个记录含有由文件 ID 识别的文件对象的属性。属性记录的内容由主机系统界定，且不由存储器系统解译。ATTR 中针对每个文件对象存在一个条目，主机已经写入所述文件对象的属性。ATTR 区块中的 ATTR 记录由 DIR 条目直接寻址，且当 ATTR 页被修改时，DIR 页被修改。

ATTR 包含在一个或一个以上专用 ATTR 区块中。只有一个 ATTR 区块可含有 ATTR 记录可写入其中的未经编程的页。ATTR 区块可含有由已经写入的 ATTR 页引起的废弃页，以及由已经删除的文件引起的有效 ATTR 页内的废弃 ATTR 记录。

ATTR 区块的页以与 DIR 页中的条目相同的次序，含有一组由同一 DIR 页内的 DIR

条目参考的 ATR 记录。一个 DIR 页可参考多个 ATR 页。因此，可对 ATR 页进行修改，随后只需要修改单个 DIR 页。

ATR 页标头存储对 ATR 页与其相关联的 DIR 页的参考，以及 ATR 页内的 ATR 记录信息的长度。ATR 页标头还可存储 ATR 页被写入时 ATTR 区块的每一者中存在的废弃页的数目的记录。此信息通常将只在最近写入的 ATR 页标头中有效。ATR 记录含有以下字段：

- 1) 文件 ID。
- 2) ATR 记录的长度。
- 3) ATR 记录的内容。属性记录的内容由主机系统界定，且不由 DFS 系统解译。

如上文所论述，数据群组是一组在文件内具有邻接逻辑偏移地址、在单个区块中的邻接物理地址处编程的文件数据。常将文件编程为多个数据群组。数据群组可具有一个字节到一个区块之间的任何长度。每个数据群组优选编程有一个标头，其含有数据群组所属的文件的文件 ID。

FIT 记录包含在一个或一个以上 FIT 区块中，且由 DIR 条目直接寻址。只有一个 FIT 区块将含有可用于对新的或经更新的 FIT 记录进行编程的经擦除页。在所有其它 FIT 区块中，所有页都已经被编程，但区块可含有完全废弃或部分废弃的页。

通过以下方式来执行对由废弃 FIT 记录占据的此容量的回收：将一个 FIT 区块指定为下一个要回收的区块，且在擦除所述被回收的区块之前，逐渐地将来自此回收区块的页复制到当前由 FIT 写入指针指定的页。

当先前对 FIT 区块进行的回收过程已经完成且所述区块已经被擦除时，选择回收区块。选择具有最高数目的废弃页的 FIT 区块作为回收区块。针对每个 FIT 区块在最近写入的 FIT 页的 FIT 页标头中记录此值。含有 FIT 写入指针的 FIT 区块不应被选择为用于回收。选定的 FIT 区块保持作为回收区块，直到回收过程已经完成且所述区块已经被擦除为止。

回收含有废弃页的 FIT 区块的过程需要以调度的时间间隔在个别突发中将含有有效 FIT 记录的少量页从所述区块复制到由 FIT 写入指针指定的页。一个突发中的页的数目应是元页中所含有的页的数目。对 FIT 写入指针处的页进行编程应作为对元页的单个编程操作来执行。

应以由 FIT 写入指针通过 4 个元页中所含有的所述数目个页位置的进程界定的时间间隔来调度回收区块中页的突发复制操作。当突发复制操作被调度时，写入指针指向物理元页中的第一个页。

可单独地对 ATTR 区块应用同一回收过程。

实施例变化形式

在上文所述的两个特定文件索引实施例的每一者中，来自主机的数据的页、FIT 条目和目录条目都存储在一个或一个以上区块的单独群组中。作为此布置的第一变化，含有目录条目的页和含有 FIT 记录的页可一起存在于同一区块中。所述区块中的多个逻辑页含有目录条目，如图 20 中所示。特定文件 ID 的目录条目在所述区块内具有已知逻辑地址。每个逻辑页还含有一组逻辑页指针，如图 21 中所示。这些充当区块中的目录页的逻辑到物理页映射。针对区块中的每个逻辑页存在一个指针，且所述指针指定逻辑页的当前物理页地址。含有 FIT 记录的 FIT 页也可存在于所述区块中。不将 FIT 页指派给逻辑页，而是在物理页位置处由目录条目直接寻址 FIT 页。当 FIT 页中的一个或一个以上 FIT 记录需要更新时，从相关目录条目识别 FIT 页的位置，读取所述页，修改相关记录，且在所述区块中的下一可用未经编程的页处写入所述经更新的页。为了更新目录条目以直接寻址此新的 FIT 页位置，单个目录逻辑页被修改且在由未经更新的 FIT 页占据的物理页位置之后的物理页位置中编程。此目录页的逻辑页指针还在同一操作中被更新和编程。所述区块中的所有逻辑页的逻辑页指针只在区块中最后写入的目录页中有效，所述目录页总是区块中最后写入的页。

或者，FIT 页还可含有所述组逻辑页指针，且 FIT 页可以是区块中的最后写入的页。

在两个特定文件索引实施例中所述的布置的又一变化形式中，含有目录条目的页、含有 FIT 记录的页以及含有来自主机的数据的页可一起存在于同一区块中。当数据从主机写入时，其以数据群组的形式在区块中的多个邻接物理页中编程。此数据群组的位置由经更新的 FIT 页中的相关 FIT 记录中的字段识别，所述经更新的 FIT 页在紧跟在数据群组的结尾之后的物理页中编程。目录页也紧跟在 FIT 页之后被编程，以更新指向经更新的 FIT 页中的所有 FIT 记录的指针，如上文所述。

数据群组不应延伸超过区块中的倒数第三个页，以允许 FIT 页和目录页紧接在其后被编程。

总结

尽管已经相对于本发明的示范性实施例描述了本发明的各个方面，但将了解，本发明有权在所附权利要求书的整个范围内受到保护。

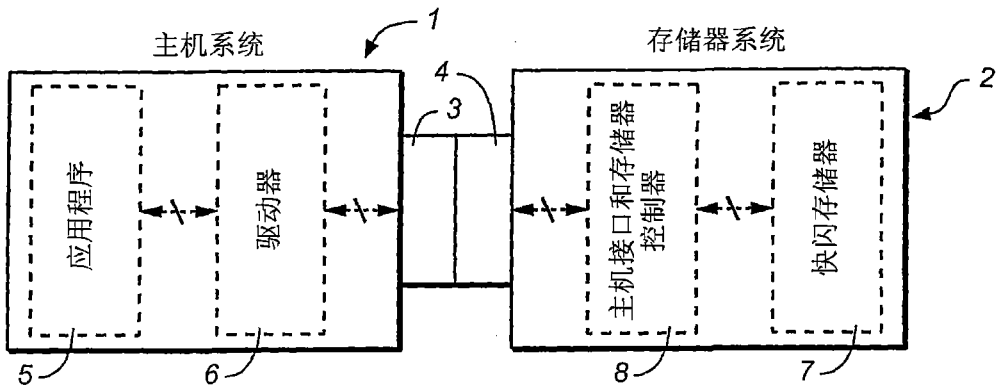


图1

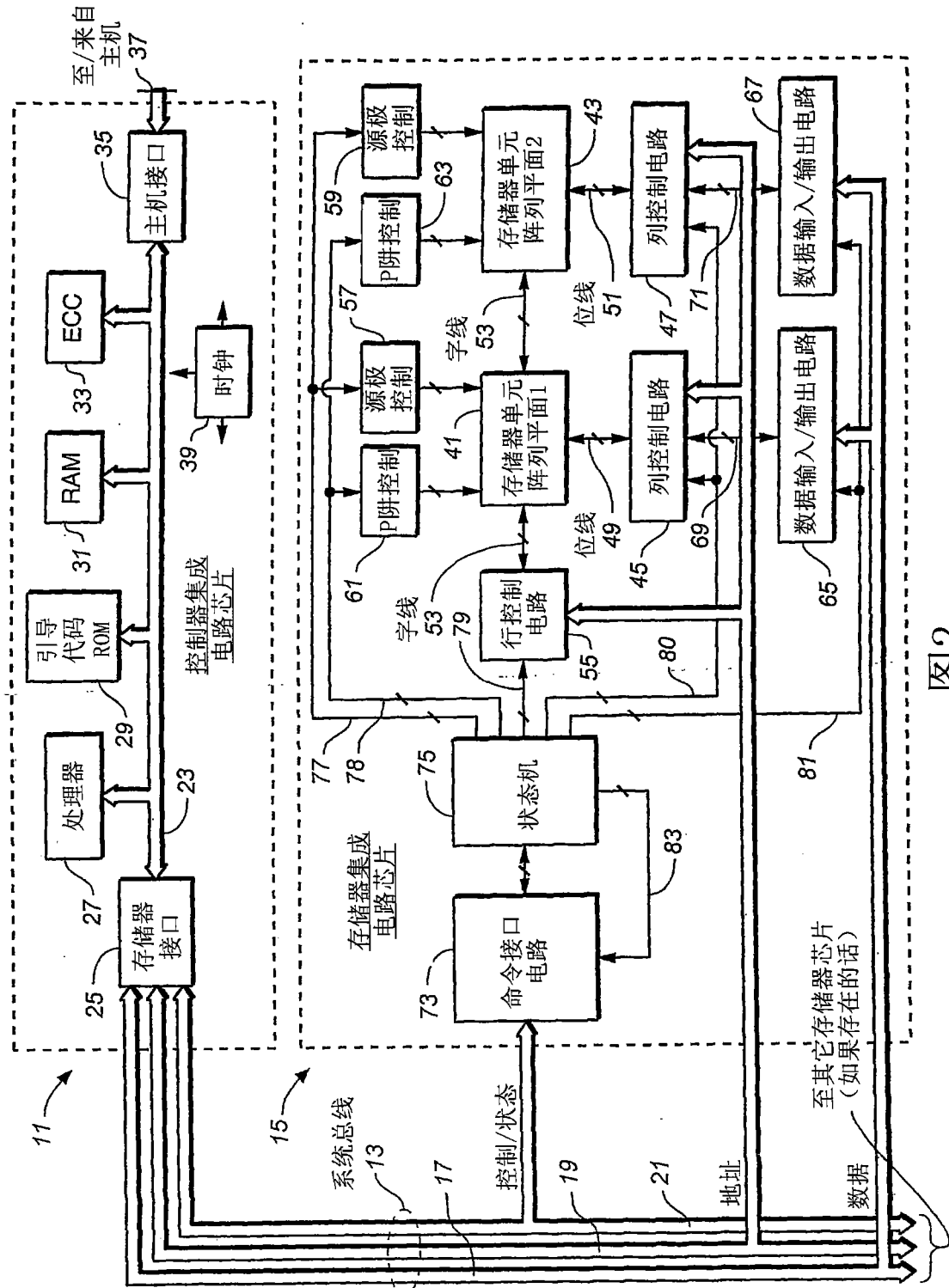


图2

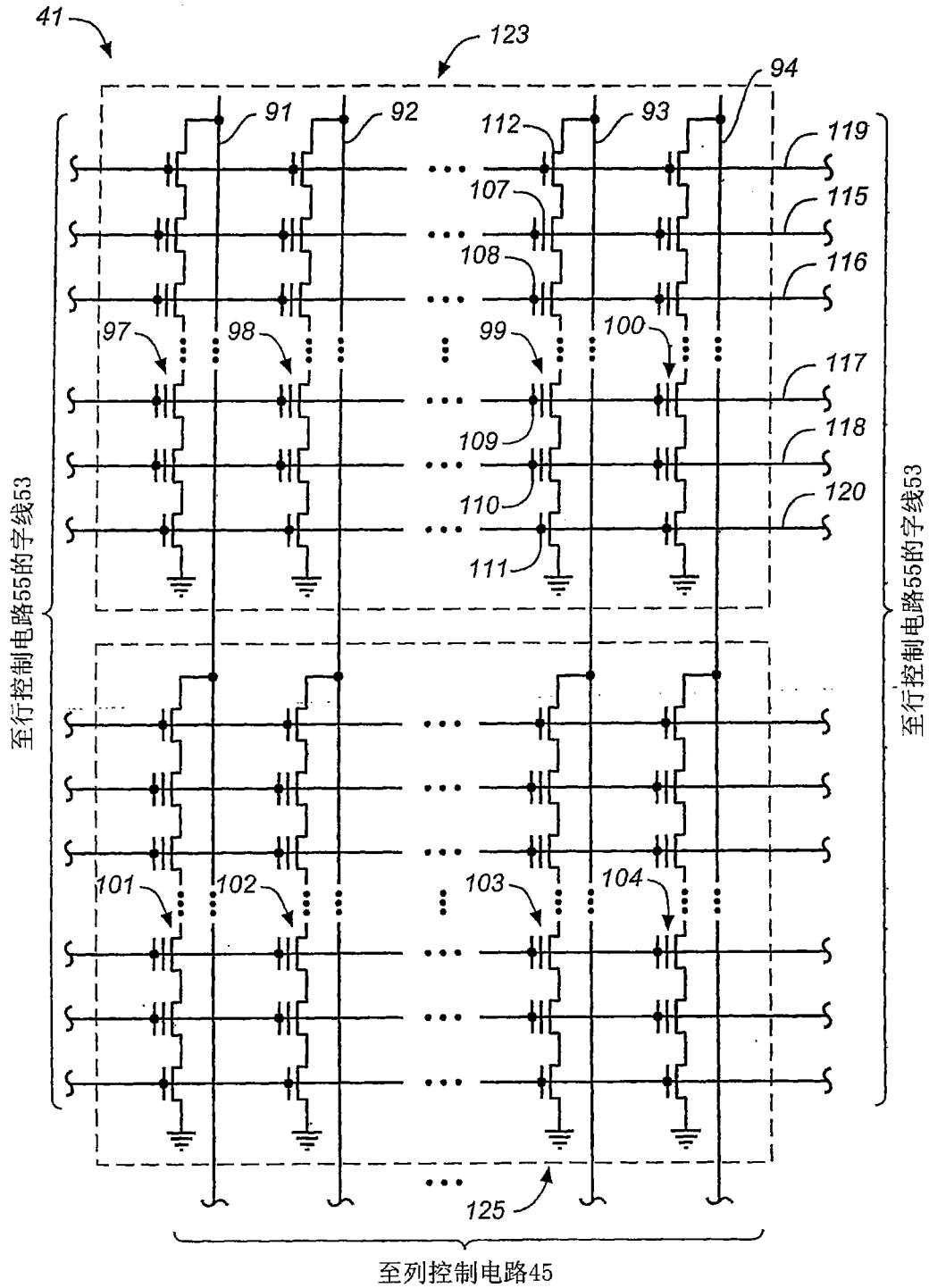


图3

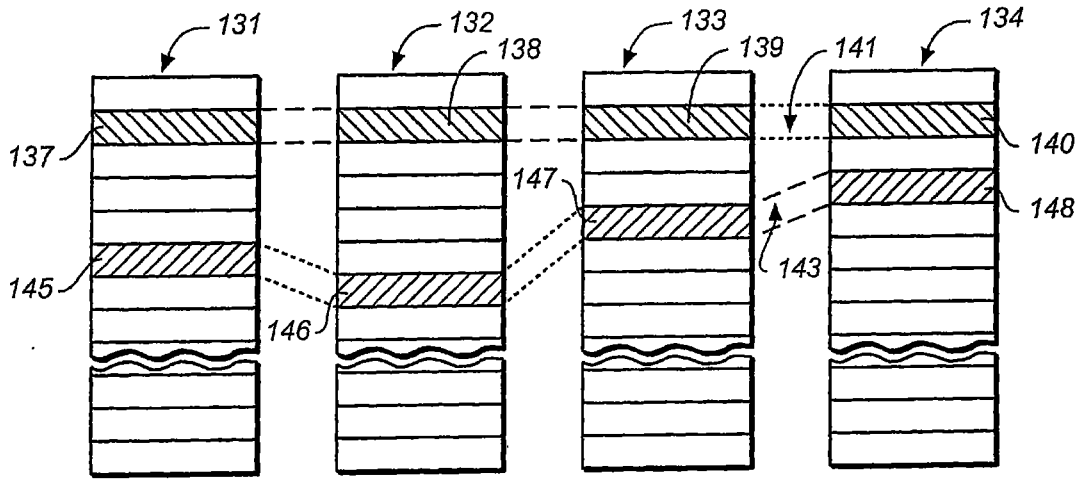


图4

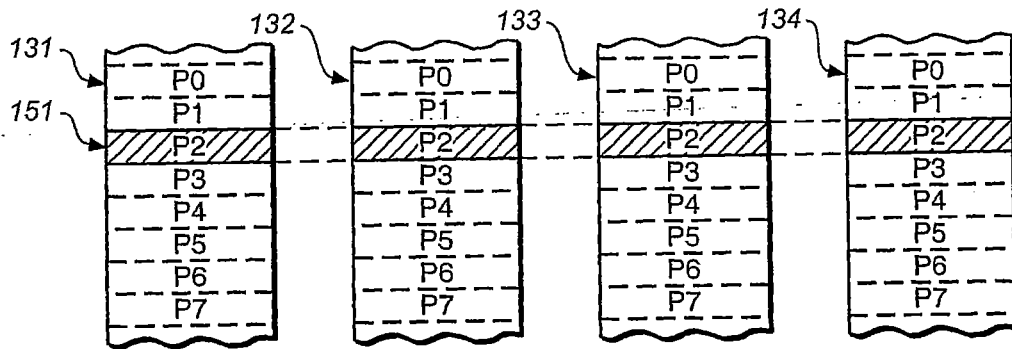


图5

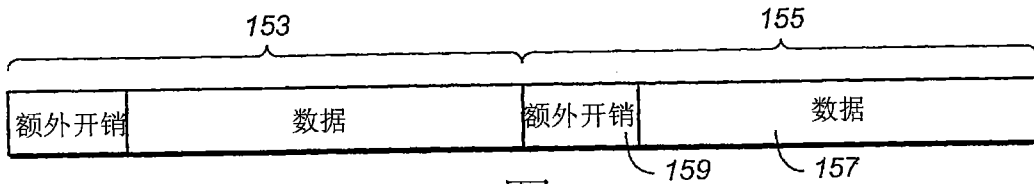


图6

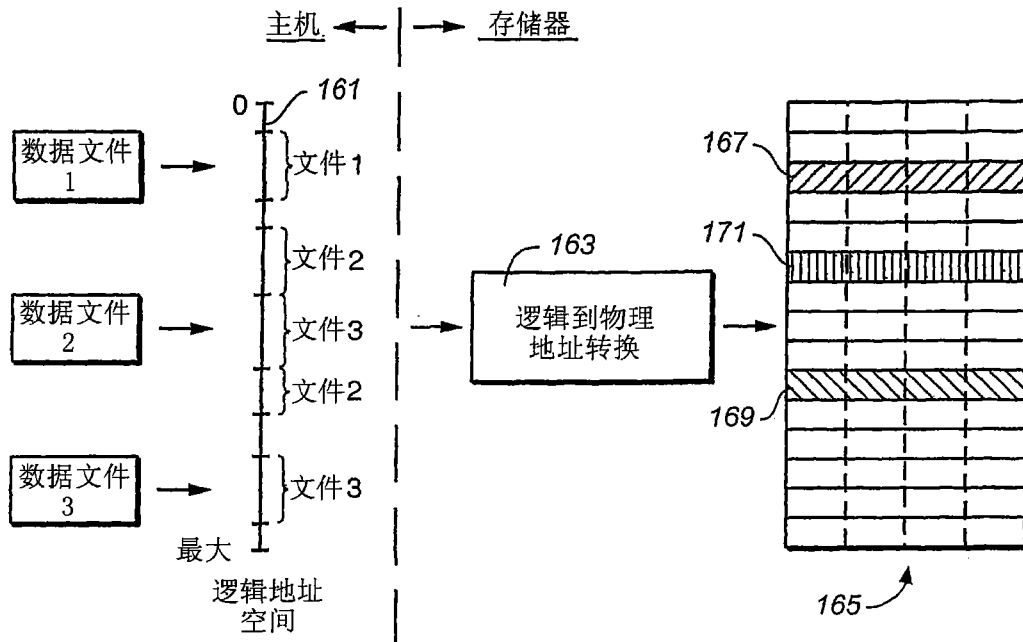


图7

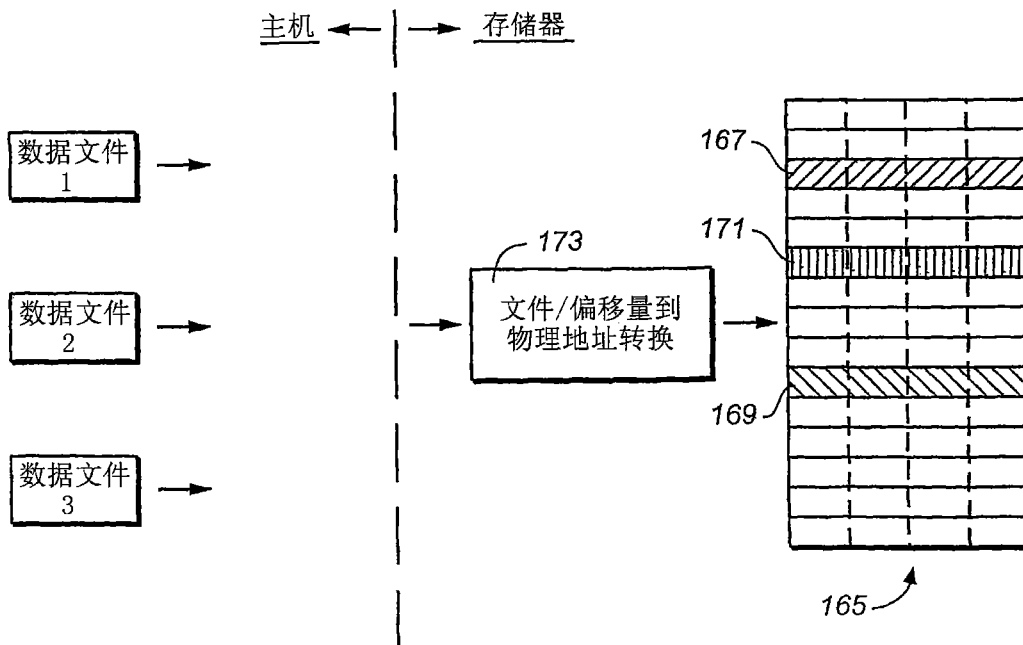


图9

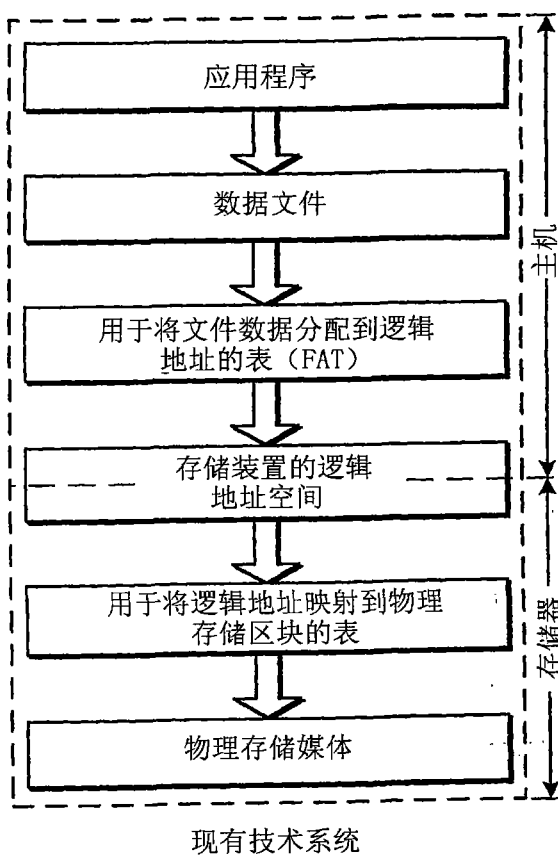


图8

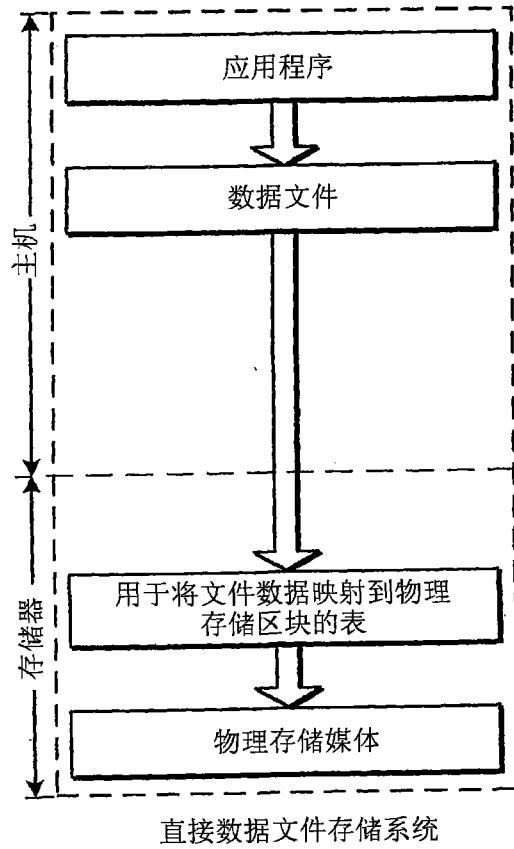


图10

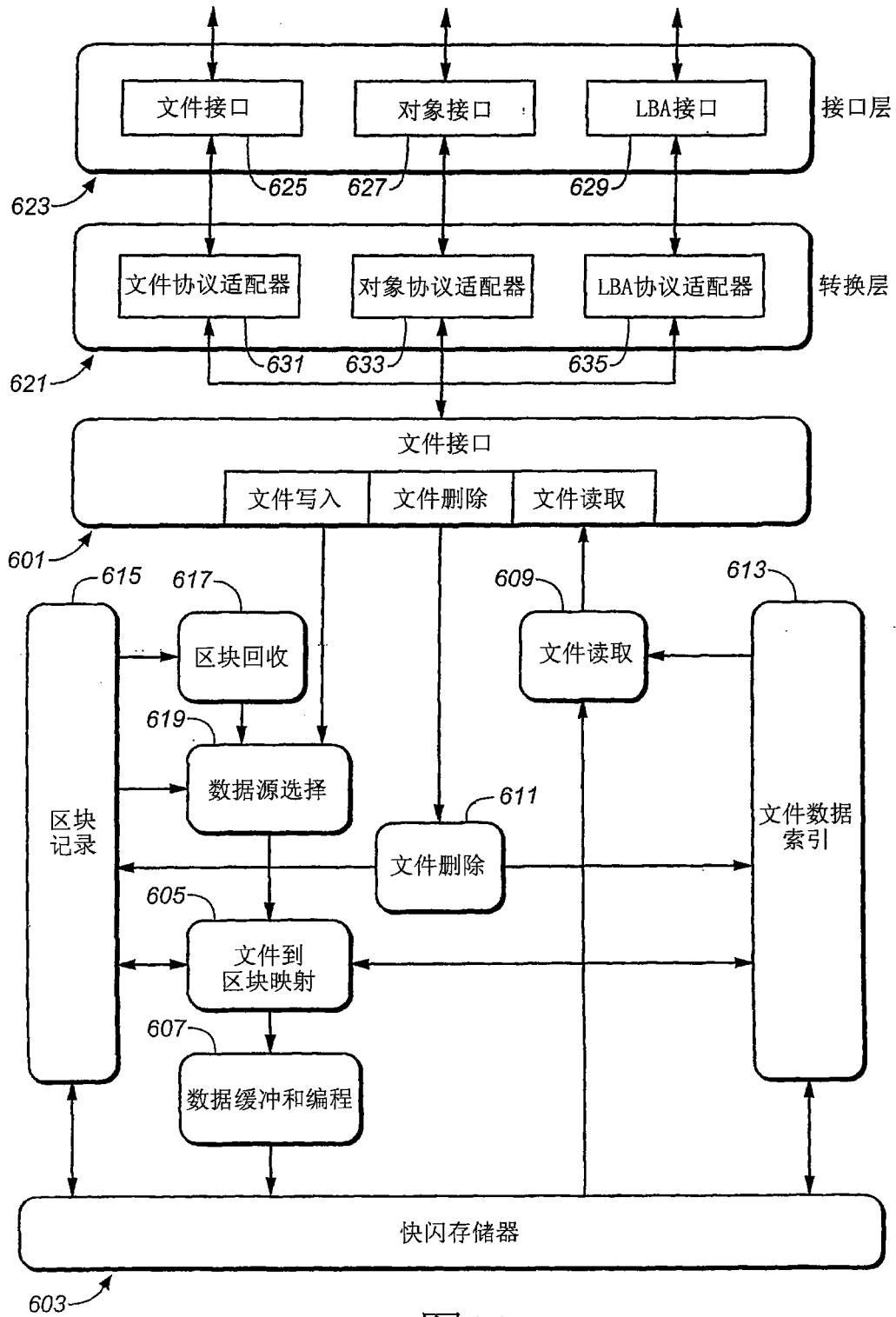


图11

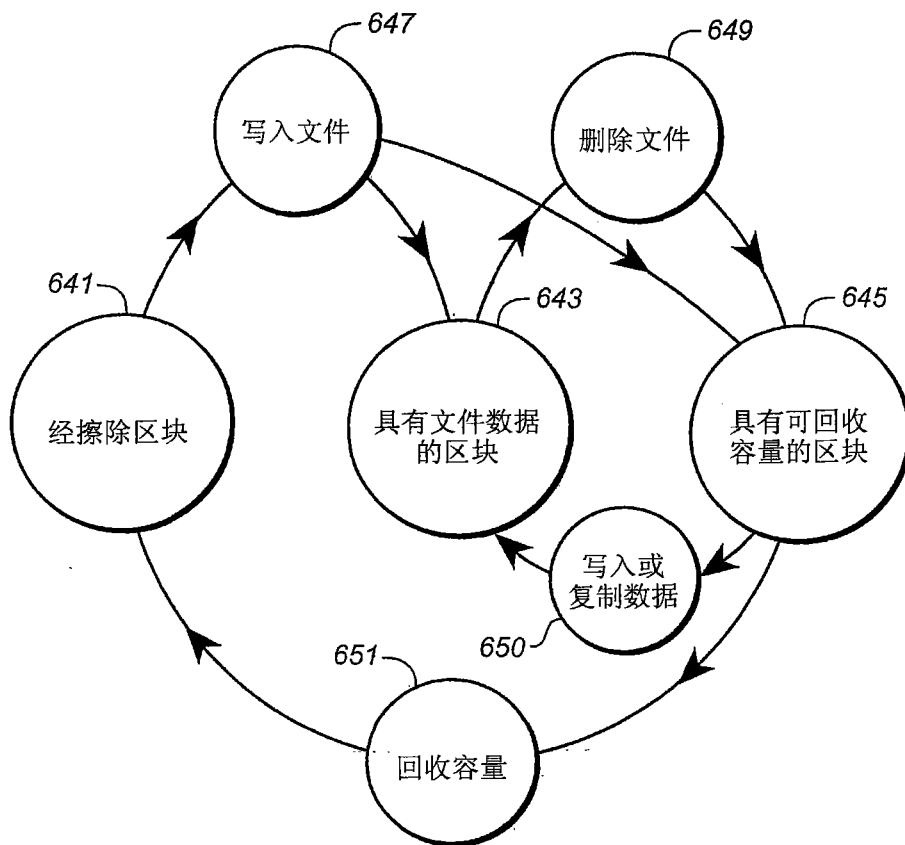


图12

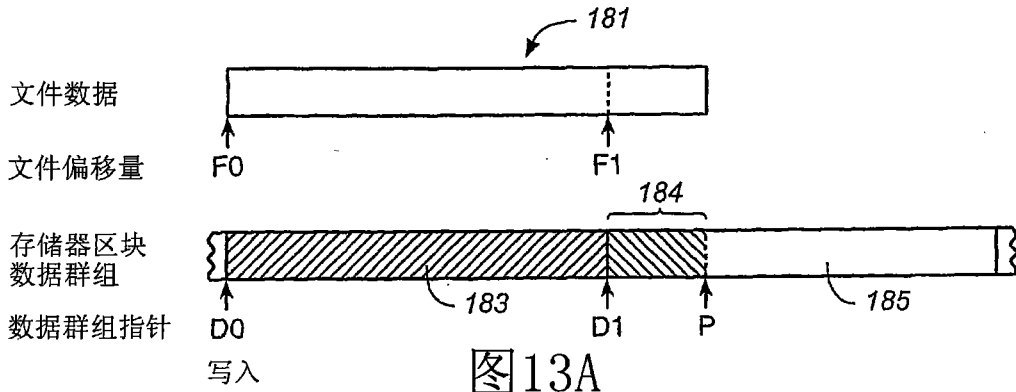


图13A

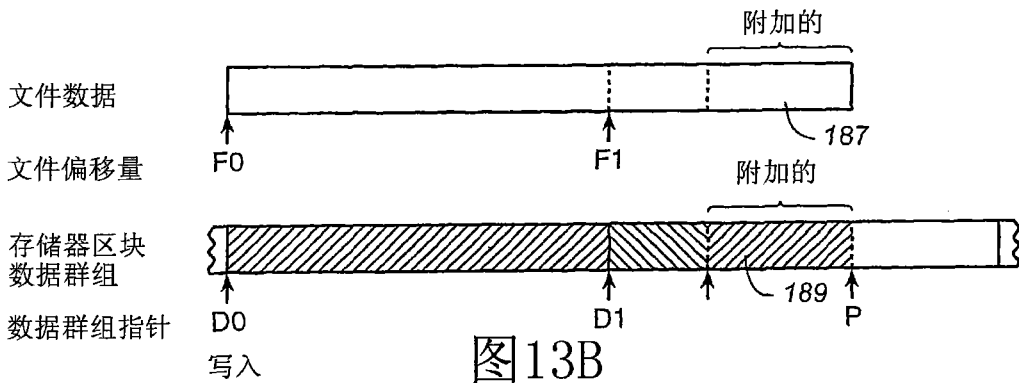


图13B

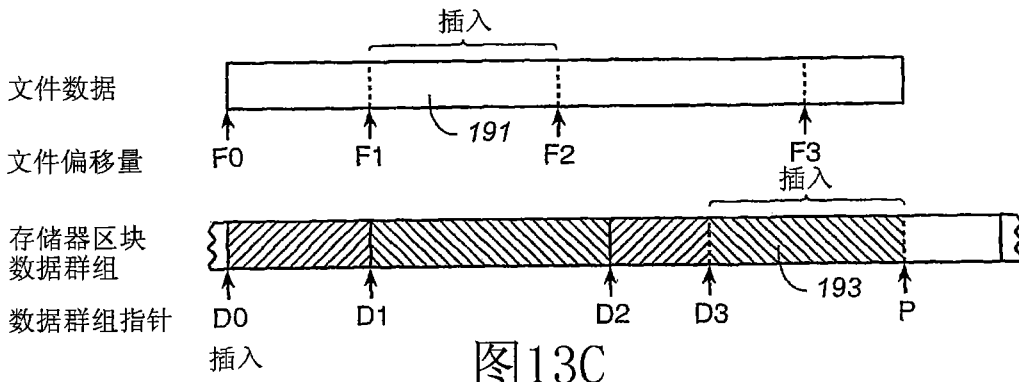


图13C

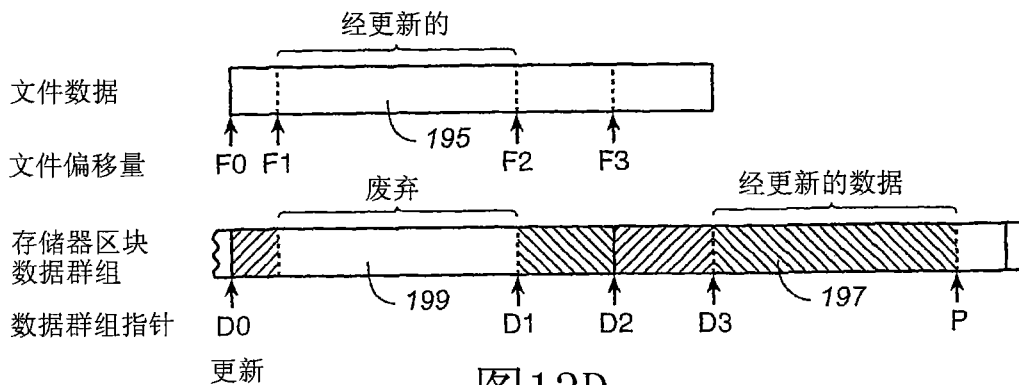


图13D

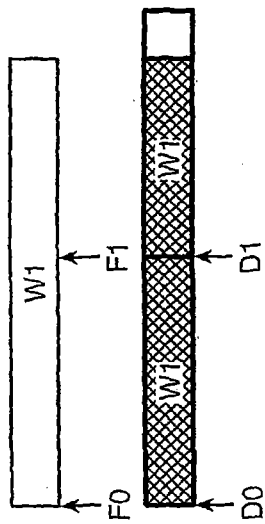


图14A

写入

文件数据
 文件偏移量
 数据群组 and 快闪区块
 数据群组指针

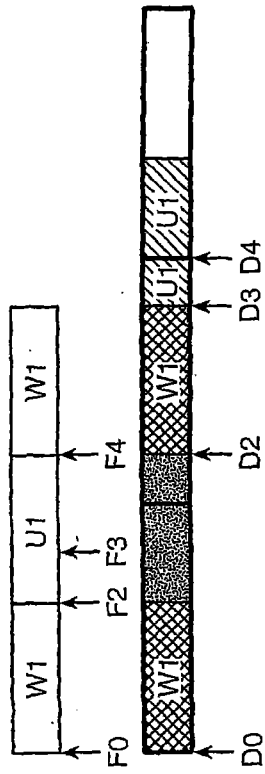


图14B

更新

文件数据
 文件偏移量
 数据群组 and 快闪区块
 数据群组指针

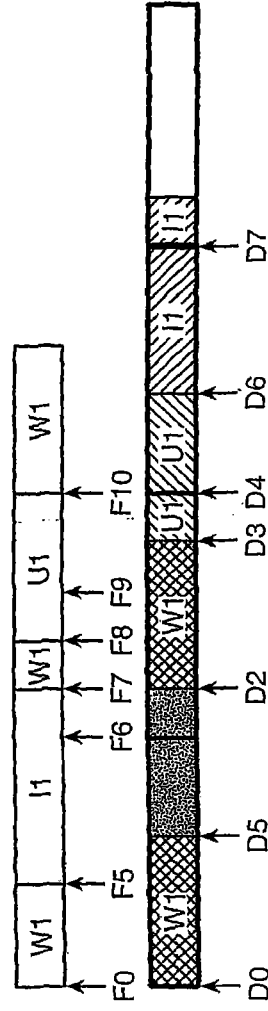


图14C

插入

文件数据
 文件偏移量
 数据群组 and 快闪区块
 数据群组指针

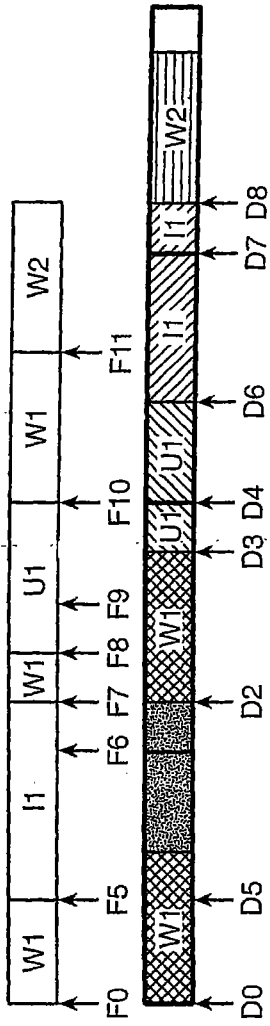


图14D

更新

文件数据
 文件偏移量
 数据群组和快闪区块
 数据群组指针

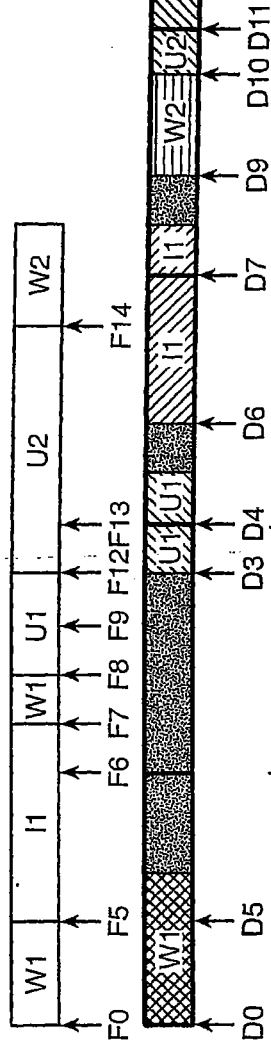


图14E

文件数据
 文件偏移量
 数据群组和快闪区块
 数据群组指针

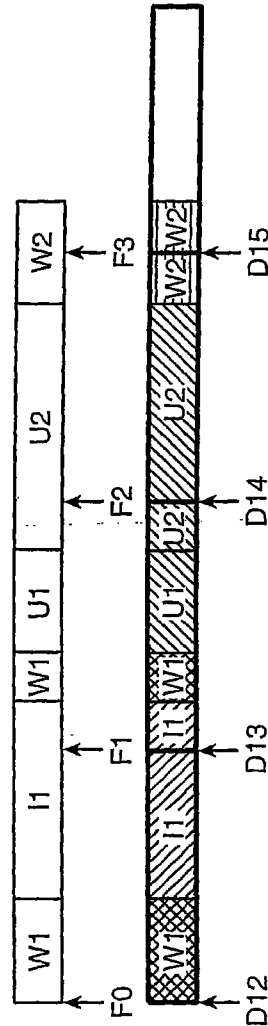


图15

文件数据
 文件偏移量
 数据群组和快闪区块
 数据群组指针

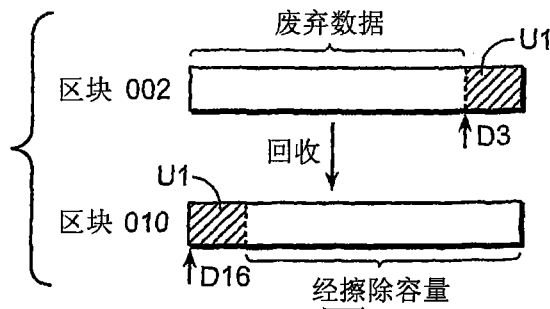


图16

文件索引表 (FIT)

	偏移量	区块	字节	长度
时间 0 (图14A)	F0	001	D0	——
	F1	002	D1	——
时间 2 (图14C)	F0	001	D0	——
	F5	003	D6	——
	F6	004	D7	——
	F7	001	D5	——
	F8	002	D3	——
	F9	003	D4	——
时间 4 (图14E)	F10	002	D2	——
	F0	001	D0	——
	F5	003	D6	——
	F6	004	D7	——
	F7	001	D5	——
	F8	002	D3	——
	F9	003	D4	——
时间 X (图15)	F12	004	D10	——
	F13	005	D11	——
	F14	004	D9	——
	F0	006	D12	——
时间 Y (图16)	F1	007	D13	——
	F2	008	D14	——
	F3	009	D15	——
	F0	001	D0	——
	F5	003	D6	——
	F6	004	D7	——
	F7	001	D5	——
	F8	010	D16	——
	F9	003	D4	——
	F12	004	D10	——
F13	005	D11	——	
F14	004	D9	——	

图17

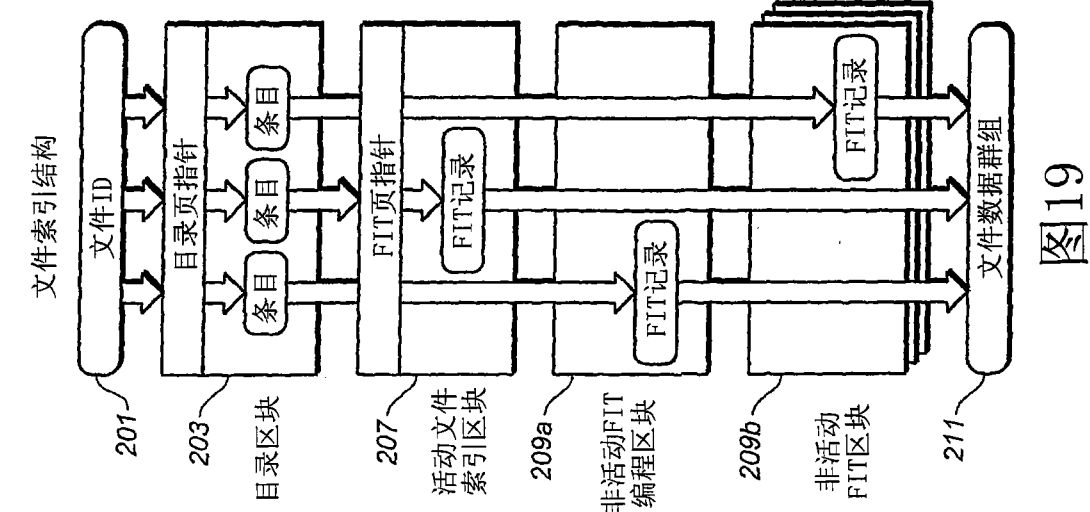


图19

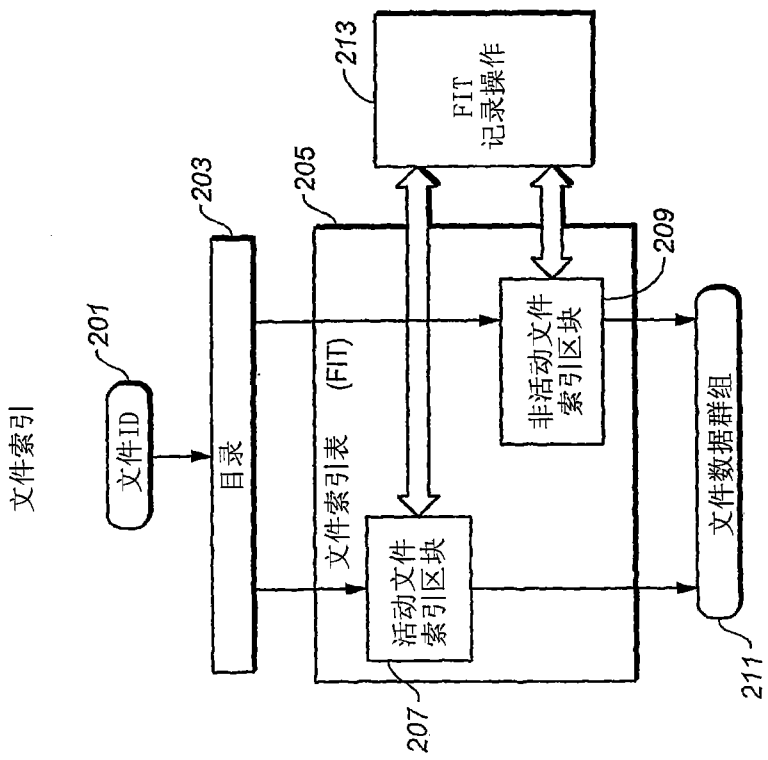
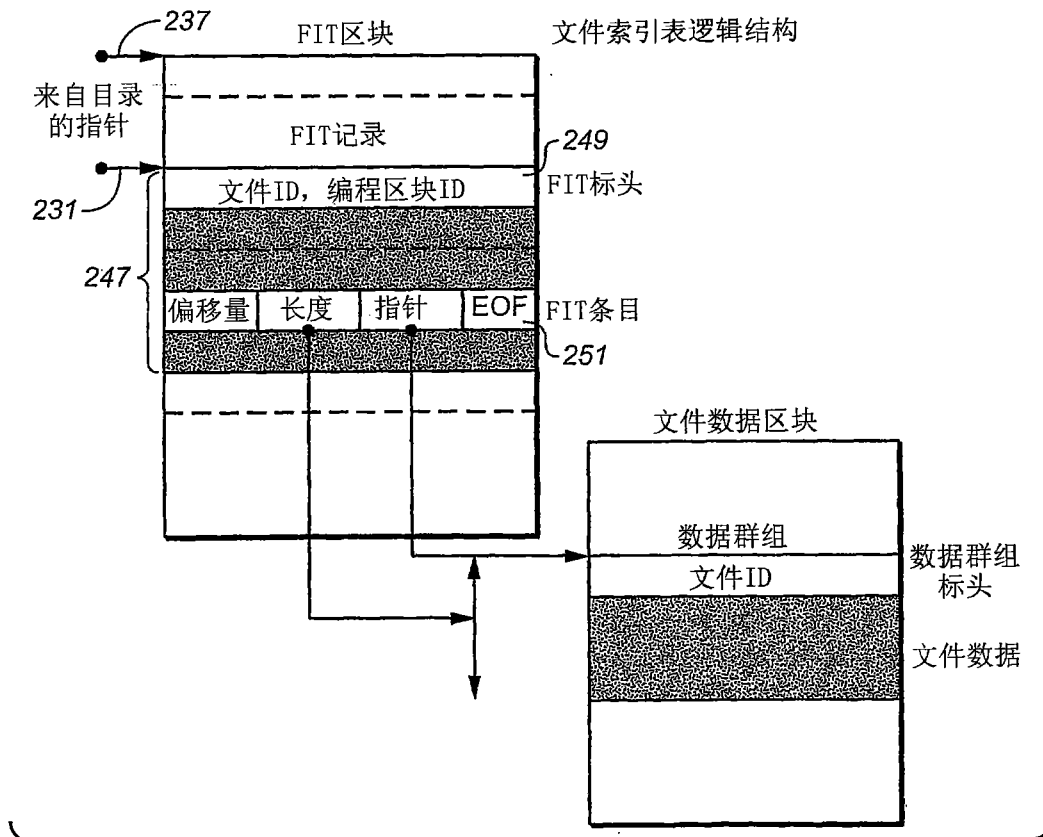
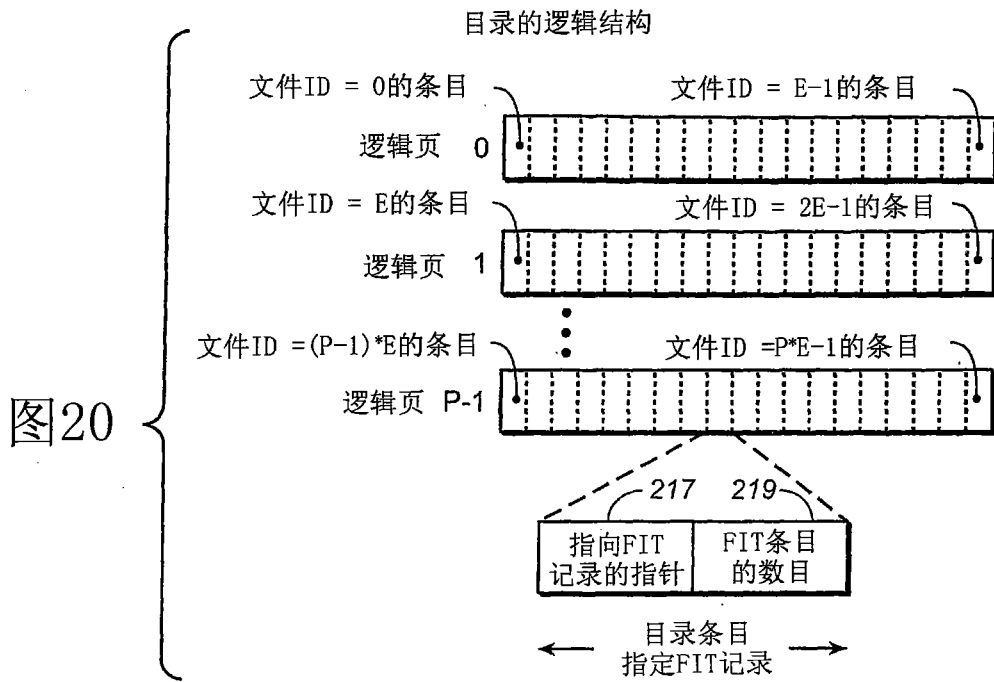


图18



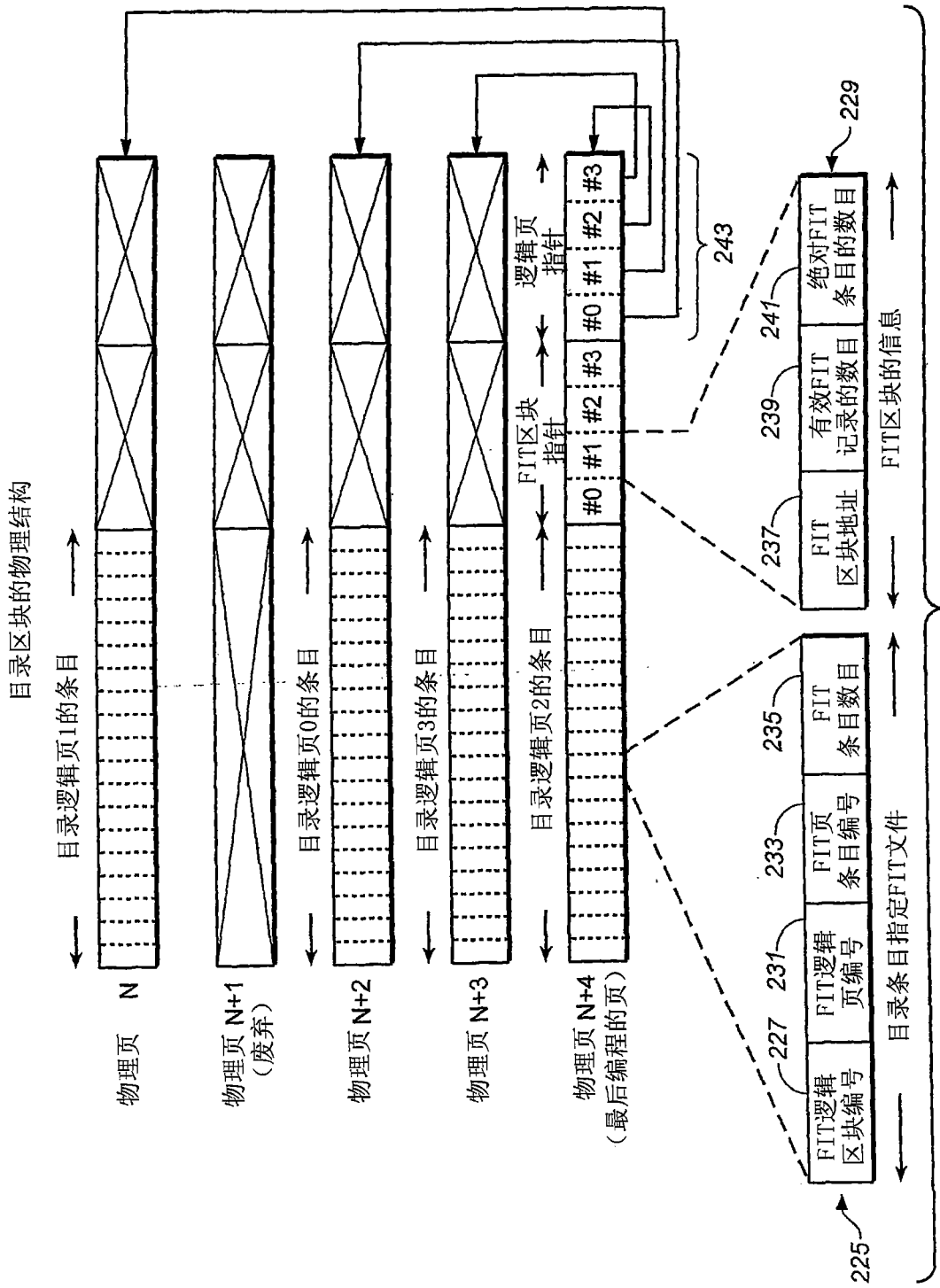


图21

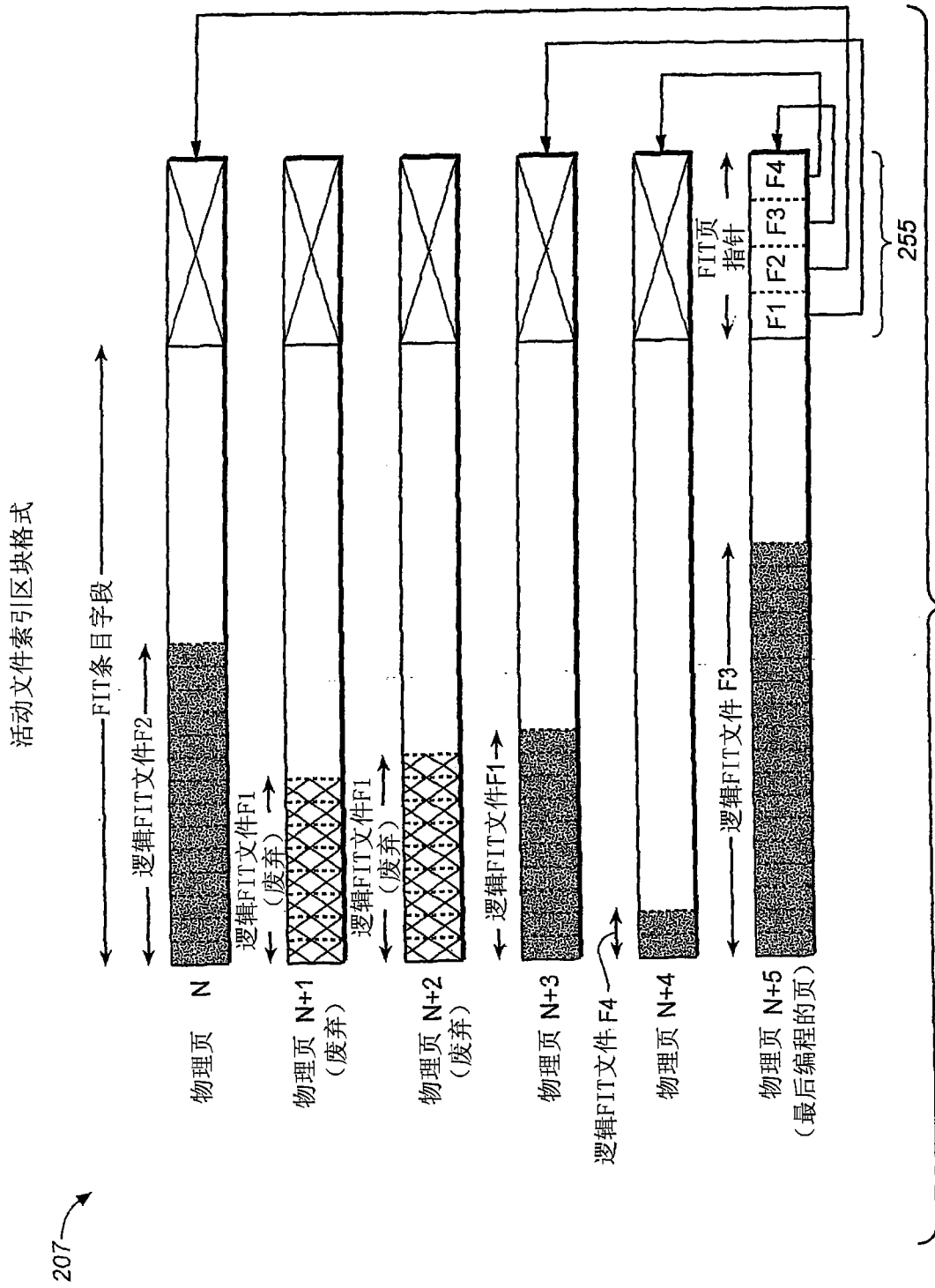
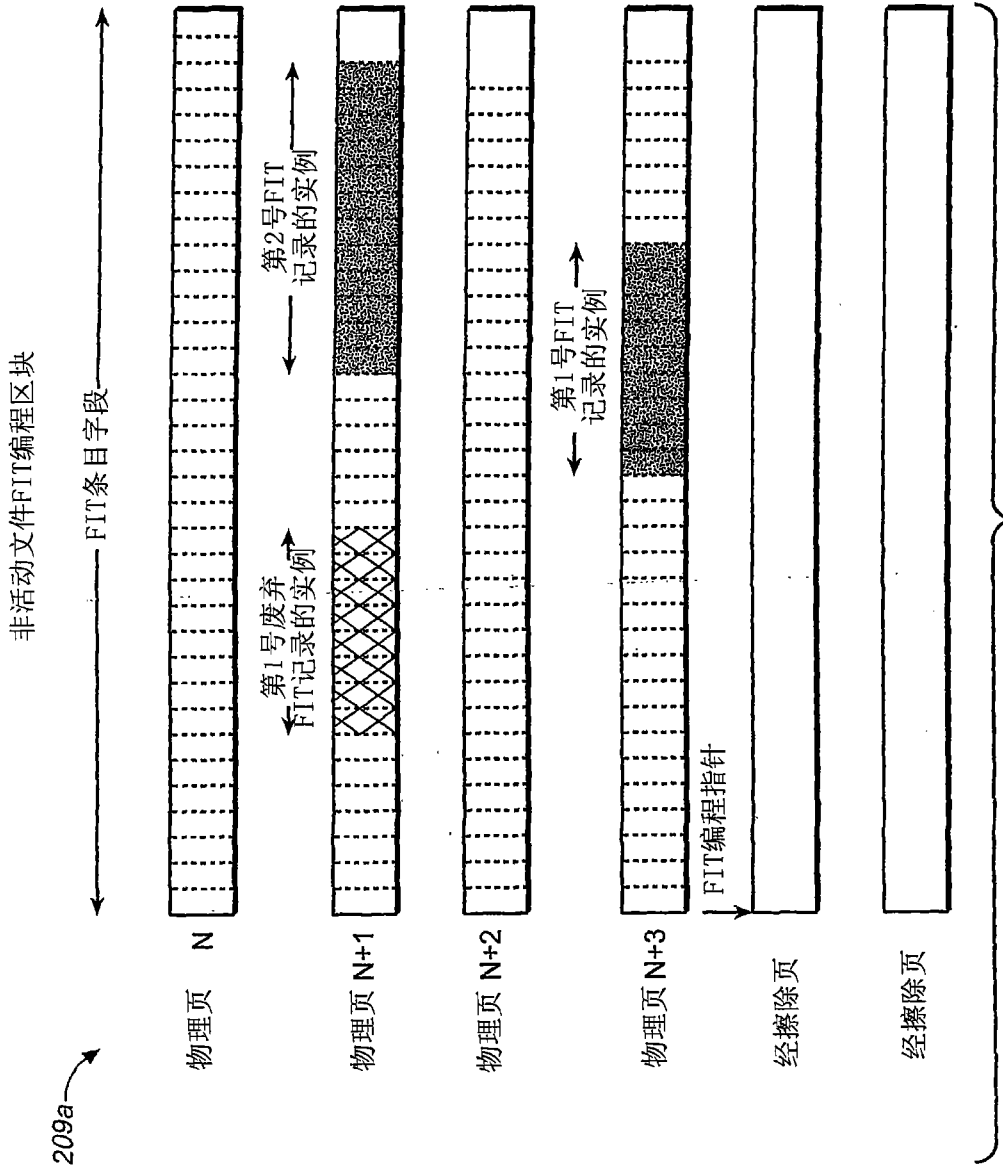


图23

207



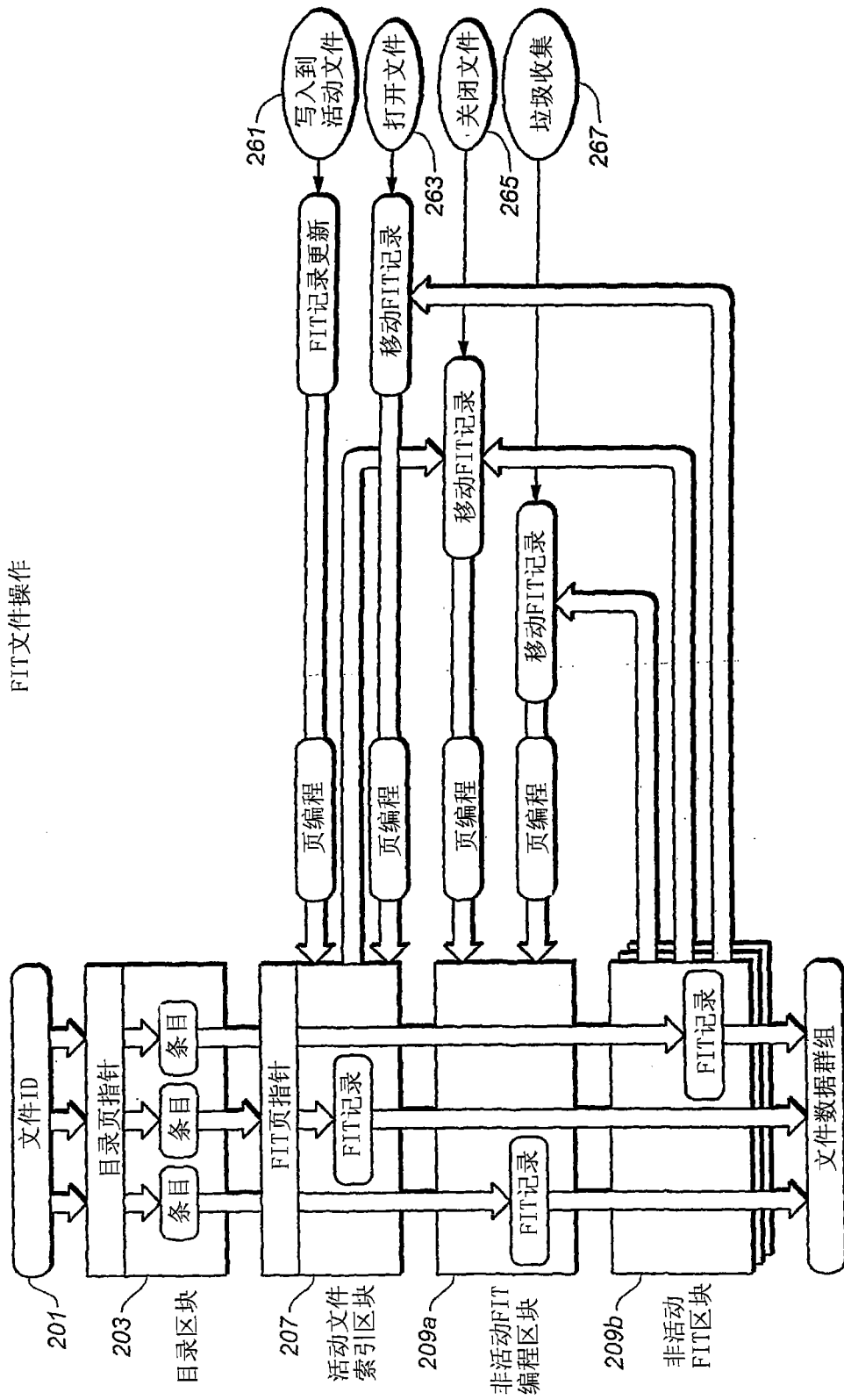
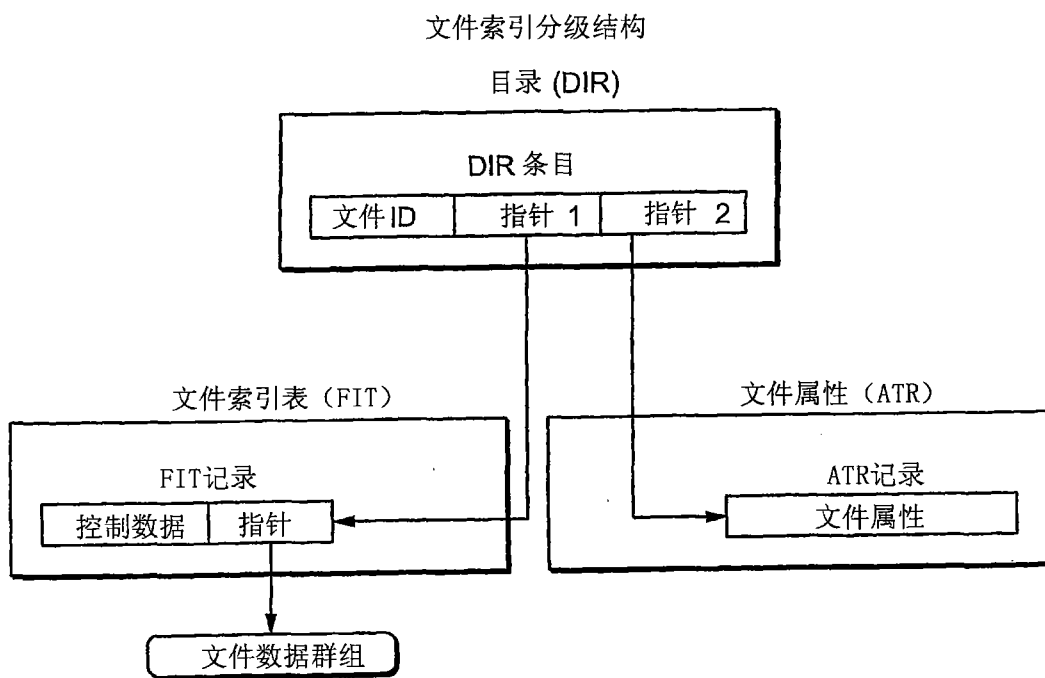


图25



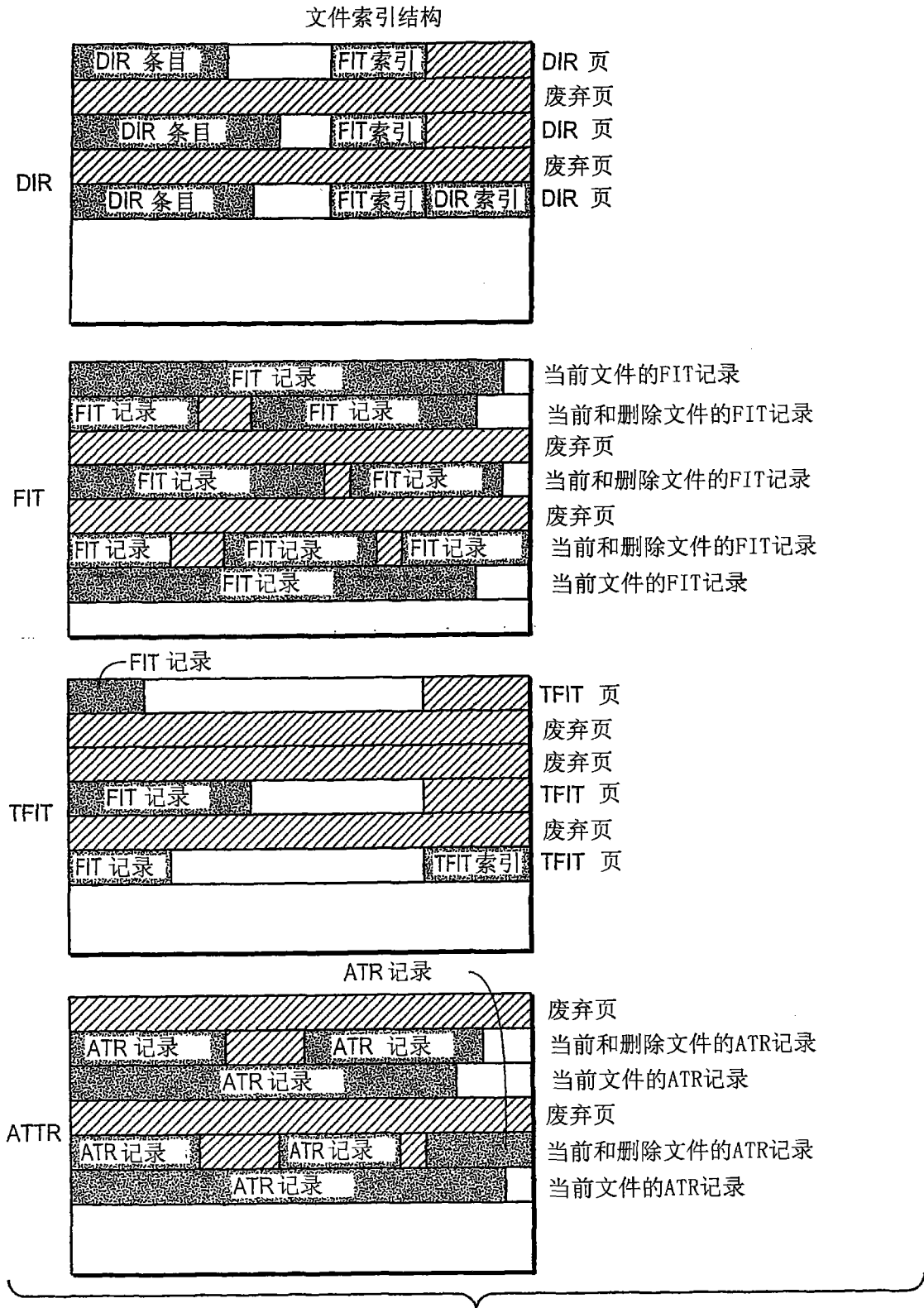


图27